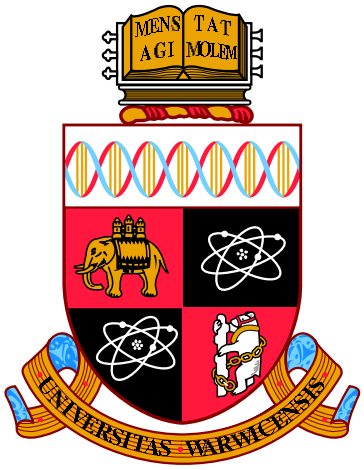


Thermalisation and Memory in Crystal Nucleation

by

Craig Devonport

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Physics

December 2020

Contents

List of Tables

List of Figures

Acknowledgments

Firstly, I would like to thank my supervisor, David Quigley, without whom I would not have started a PhD let alone written this thesis. I am extremely grateful for all the support and guidance he has provided. He has allowed me the freedom to explore the areas that interested me, while keeping a close enough eye to dissuade me from any foolhardy ventures.

I would also like to thank the scientific computing RTP staff for administering pretty much all of the machines I used to undertake this research. Without their work my days would have been filled with more cursing at computers than they already contained.

Finally, I would like to thank all of the friends I've made during my time at Warwick. There are too many of you to reasonably name, but you have all had an influence on me, though I cannot say whether this influence has always been a good one or not. Thank you for a lovely game of PhD.

Declarations

All work in this thesis is original work conducted by the author with the exception of the 2D Ising model umbrella sampling and free energy fitting, section 5.2 table 5.1 and figure 5.10, which was conducted by Dr Dipanjan Mandal.

Abstract

In this thesis, we examine two assumptions made in classical nucleation theory. Those being, the cluster size n is a slow degree of freedom and all other degrees of freedom thermalise on more rapid timescales, and that the dynamics of that slow degree of freedom are well described by a Markovian random walk on the resulting free energy landscape. We examine the quasi-equilibrium assumption using a substantial set of Lennard-Jones simulations and plotting the velocity distribution of the particles for each n sampled over the course of a seeded nucleation trajectory. Comparing these distributions we find that all degrees of freedom other than n are in thermal equilibrium, provided these are sampled from both trajectories which include growing and shrinking clusters. The consequences of this for calculations of the nucleation rate via variants of seed methods are discussed.

To examine the Markovian assumption we take a maximum likelihood approach to score trajectories against reference stochastic process based on two memory models, one with a linear memory function and one with a reciprocal length function. Applying these models to molecular dynamics seeding style trajectories we find that the trajectories are best described by allowing for steps in n greater than 1 and either a short non zero memory (length 1 to 3) with a linear memory function or simply a non zero memory (length 1 to 10) with the reciprocal memory function.

In the final chapter we explore the feasibility of using neural network models to predict the committor for nucleation via magnetisation reversal in the 2D Ising model. These predictions are calculated from from collective variables as well as from the current grid state of the simulation and perform better than a polynomial regression fitted using two collective variables, the size of largest cluster n and the total number of spin up site in the grid $N_{\uparrow}$.

Chapter 1

Introduction

Crystal nucleation is important for studying phase transitions in materials, these transitions are of great importance to a diverse range of fields. These fields range from the ecological impact of climate change which is influenced by ice formation [3], to the concerns of the oil industry where gas hydrate build up and crystallisation can have a major impact on oil extraction and transport [4, 5]. The industrial concerns are not limited to the oil industry as drug design conducted by pharmaceutical companies requires generating the correct polymorph which is highly impacted by the early stages of crystallisation [6]. Some areas of medicine are also concerned with crystal nucleation, as a better understanding could reduce the production of kidney stones [7] or even help fight neurodegenerative disorders such as Alzheimer's [8, 9].

We would like to be able to use simulations to examine what is happening on time and length scales shorter than those which are accessible by practical experiments. By examining these processes we hope to use simulation to design mechanisms to control, inhibit or promote nucleation, i.e. control the rate at which a metastable parent phase transforms into a more stable phase. To compare simulation to the physical processes they are trying to model, we need to calculate some macroscopic quantity which can also be extracted from practical experiments. For the case of nucleation process and obvious choice would be the nucleation rate.

Unfortunately, the time scales on which nucleation occurs can be out of reach for computational simulation of all but the simplest systems. To combat this several methods have been developed to focus the computational effort of simulation to the nucleation pathway. One of these approaches is the class of methods referred

to as seeding methods. These methods aim to avoid unnecessary computation by simulating from preformed seeds. By describing the time evolution of these seeds using classical nucleation theory (CNT), these simulations can be used to calculate nucleation rates, and depending on the seeding variant used, other parameters of interest.

As these methods use CNT, they only work for systems in which CNT is valid. This requires that the free energy is well described by the chosen reaction coordinate, i.e. the reaction coordinate is a good description of the nucleation pathway. This reaction coordinate is typically taken as the size of a growing cluster of crystalline particles. Furthermore, it requires that the dynamics of the system are well described by a Markovian random walk on that free energy landscape. For this to be the case, all degrees of freedom in the system other than the cluster size must come to a quasi-equilibrium state at each cluster size before the cluster size changes, i.e. the reaction coordinate is a slow degree of freedom.

While these seeding methods, along with other methods, allow nucleation rates to be calculated from simulations, results from these calculations often vary from experimental results by orders of magnitude even for relatively simple systems such as ice nucleation [10] and colloidal crystallisation [11]. Given the size of the discrepancies, any simulations studying nucleation control can only hope to offer suggestions based on qualitative trends where there are large changes in nucleation rate. The difference in results between simulations and experiments is mostly attributed to two main causes, the accuracy of the force field used to simulated interactions, and the assumptions made in the method used to calculate the nucleation rate. There is clearly a need for improved improved tools for calculating nucleation rates. These could be improvements to force field accuracy or improvements to the methods used. If we wish to improve existing methods we should examine their assumptions.

In this thesis, we examine some of the assumptions made by seeding methods. We will begin by introducing some relevant background and give a brief overview of some related literature in chapters 2 and 3. We will then discuss the implementation of two forms of seeding method for Lennard-Jones systems in chapter 4 and examine the thermalisation of and heat transport within this Lennard-Jones system. Next, in chapter 5, we will discuss a method of identifying and quantifying the effect of different forms of memory in nucleation trajectories to test the assumption that the cluster size dynamics are Markovian. To do this we will begin by testing the memory of known non-Markovian walks against reference stochastic process, then

apply the method to dynamics of cluster size during magnetisation reversal in the 2D Ising model as well as molecular dynamics trajectories of nucleation from the melt. We will then discuss a method of approximating the committor, the ideal reaction coordinate, for the 2D Ising model using machine learning techniques in chapter 6. Finally we will conclude in chapter 7.

Chapter 2

Background

In this chapter we will introduce some relevant theoretical background relied on in the following chapters.

2.1 Markov Processes

A Markov process, also referred to as a Markov chain, is a stochastic process in which the probability of any event happening only depends on the current state. In these processes, the time period between events or state changes can be either discrete or continuous, both of which have a multitude of uses in various fields. In discrete time models, each step represents a fixed amount of time, and a state change happens at each step. In the continuous time model the time between step is randomly drawn from a distribution based on a decay rate or other factor associated with the current state. For this work we need only consider discrete time Markov processes.

2.1.1 Random Walks

In a random walk, we use random steps to explore some environment. The simplest form to consider is a 1D random walk which can only move in one of two directions, positive and negative. In this simple model, a coordinate is adjusted by increments of ± 1 based on some probability, $P(+)$ and $P(-)$ most commonly set to $P(+) = P(-) = \frac{1}{2}$. For the simplest systems, these probabilities can be set as constants which leads to an unbiased walk over all of the coordinate domain.

These random walks can be modified by adding some potential energy landscape $E(x)$ biasing the walk. In a biased random walk, the walker chooses to move

in the positive or negative direction with probabilities, $P(+)$ and $P(-)$, as in the unbiased walk, but then chooses to accept or reject that move based on the potential difference between the current (x_0) and chosen (x_1) positions, $\Delta E = E(x_1) - E(x_0)$.

In this work, we will use biased random walks to model kinetic systems. In order to do this we need to ensure that our acceptance probability $P_{\text{Accept}}(\Delta E)$ obeys detailed balance. This means the probability of starting at x_0, choosing to move to x_1 and accepting that move must equal to the probability of starting at x_1, choosing to move to x_0 and accepting that move under time reversal,

$$P(x_0)P_{\text{Choose}}(x_0 \to x_1)P_{\text{Accept}}(E_1 - E_0) = P'(x_1)P'_{\text{Choose}}(x_1 \to x_0)P'_{\text{Accept}}(E_0 - E_1). \tag{2.1}$$

A common choice for the acceptance probability is the Boltzmann factor of the potential E_0 and E_1

$$P_{\text{Accept}}(E_0 \to E_1) = \frac{\exp\{-\beta E_1\}}{\exp\{-\beta E_0\}} = \exp\{-\beta \Delta E\}, \tag{2.2}$$

with β being the inverse temperature, $\beta = 1/k_B T$. This means if we set the probability of choosing positive and negative increments to be equal,we can satisfy detailed balance,

$$\frac{P_{\text{Choose}}(+)}{P'_{\text{Choose}}(-)} = \frac{P'_{\text{Accept}}(-)}{P_{\text{Accept}}(+)} = \frac{\exp\{-\beta \Delta E'_-\}}{\exp\{-\beta \Delta E_+\}} = 1, \tag{2.3}$$

as, by symmetry, $\Delta E_+ = -\Delta E'_-$.

2.1.2 The Ising Model

The Ising model is a lattice model in which each site in the grid has 2 possible states $+1$ and -1. The model was originally developed for studying ferromagnetic systems, but has since found applications in other fields, including crystal growth and nucleation. When being used to describe crystals, as we shall use them in this work, the $+1$ and -1 states are often though of as referring to solid and liquid, or solute and solvent site. These models can have an arbitrary number of dimensions, but 2D and 3D models are most common for studying crystal growth and nucleation.

In this model, any two neighbouring sites i and j have an interaction strength of J_{ij}, and each individual site i interacts with an external field of strength h_i. This

gives the Hamiltonian for any state $\boldsymbol{\sigma}$ as

$$H(\boldsymbol{\sigma}) = -\sum_{\text{neighbours},ij} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i. \tag{2.4}$$

Where σ_i is a single site in a micro state $\boldsymbol{\sigma}$. Here we indicated the first sum to be over only neighbouring sites which could also be achieved by setting $J_{ij} = 0$ for all pairs of sites which are not neighbours. By defining this sum to be only over neighbouring pairs it allows us to set J_{ij} to be a constant, typically $J = 1$. It is also common to take $\boldsymbol{h}$ to be a constant representing a uniform magnetic field.

In the Ising model when $h > 0$, the stable state is a fully spin up grid, whereas when $h < 0$, the stable state is a fully spin down grid. If we place a fully spin down grid into an external magnetic field $h > 0$, it will be in a metastable state as the interactions between neighbours will resist spins flipping. At a non-zero temperature, the system will eventually move away from the metastable state towards the more stable phase, this happens by a process of nucleation and growth, with a cluster of the more stale phase forming and expanding. For this reason the Ising model is often used to study crystal nucleation.

The Hamiltonian gives us the energy of the system, but in order to evolve a system we need some form of dynamics. The 1D Ising model can be solved analytically, however here we will discuss two numerical methods for simulating the Ising model. The two methods we shall discuss are spin flip dynamics [12] and local spin exchange dynamics (also called Kawasaki dynamics) [13], which are both Markov chain Monte Carlo methods.

Spin flip dynamics

In spin flip dynamics we select a random site and flip the spin, $+1 \to -1$ or vice versa. We then accept or reject that move based on the Boltzmann factor of the energy change, $P_{\text{Accept}}(\Delta E) = \exp\{-\beta\Delta E\}$. This process is then repeated to continue the evolution. A common output interval for these systems is a Monte Carlo sweep, which is the number of steps for which, on average, a flip is attempted at each site. In other words, 1 flip attempt per lattice site.

Local spin exchange dynamics

Local spin exchange dynamics start with, as in spin flip dynamics, by choosing a random site within the lattice. Once a site is selected, a direction is randomly chosen, and if the spin at that neighbouring site is different to the originally chosen site, they are swapped. As with spin flip dynamics, this change is accepted with a probability of the Boltzmann factor of the energy change.

One other consideration to make is what happens when we try to exchange a site over the boundary. If we use periodic boundary conditions, then the total magnetisation, i.e. the sum of all lattice sites, will remain constant. This may be desired for some applications, but if we wish to model crystal growth, then we need a way of more particles entering the system. To allow for this, when swapping over a boundary, we choose the external site to be occupied with a +1 state with the probability of the background density. This background density can be calculated by conducting a spin flip simulation with the same field, h and temperature.

2.2 Statistical Mechanics

Statistical mechanics provides a link between the microscopic properties of particles with the macroscopic thermodynamics of a system. This link is provided by considering a collection of copies of the system (ensemble), described by the same Hamiltonian, each with a unique set of microscopic properties (microstate), which when considering the collection as a whole, allows us to determine the thermodynamic properties (macrostate). Any ensemble is characterised by its partition function, from which all thermodynamic properties can be determined. Here we will discuss the ensembles which play a prominent role in this work.

2.2.1 Microcanonical Ensemble

The microcanonical ensemble describes an isolated system with a constant energy (E), volume (V) and number of particles (N). This ensemble is often referred to as the NVE ensemble. As the system has a fixed energy, all of the microstates are equally probability,

$$p = \frac{1}{W}, \tag{2.5}$$

where W is the total number of possible microstates for the system. The partition function for this ensemble is given by

$$\Omega\left(N,V,E\right)=\frac{1}{N!h^{3N}}\iint_{V}\delta\left(\mathcal{H}\left(\mathbf{x},\mathbf{p}\right)-E\right)d\mathbf{x}d\mathbf{p}, \tag{2.6}$$

where h is Plank's constant, $\mathcal{H}$ is the Hamiltonian, and δ is the Dirac delta function. The integral is taken over all $3N$ spatial coordinates, $\mathbf{x}$, and $3N$ momentum coordinates $\mathbf{p}$ within the system volume V. The $1/N!$ pre-factor comes from the indistinguishability of the particles, i.e. we can swap any particle with any other particle and it would be impossible to tell, and the h^{-3N} pre-factor comes from quantum mechanical considerations.

2.2.2 Canonical Ensemble

The canonical ensemble extends the microcanonical ensemble by considering a system coupled to a heat bath. As the system has a fixed number of particles (N), Volume (V), and temperature (T), it is refers to as the NVT ensemble. The probability of the system being in a microstate i is

$$p_{i}=\frac{1}{Q}\exp\left\{-\frac{\mathcal{H}\left(\mathbf{x}_{i},\mathbf{p}_{i}\right)}{k_{B}T}\right\}, \tag{2.7}$$

Where $\mathbf{x}_i$ is the position vector, $\mathbf{p}_i$ is the momentum vector, k_B is Boltzmann's constant, and Q is the partition function given by

$$Q(N,V,T)=\frac{1}{N!h^{3N}}\iint_{V}\exp\left\{-\frac{\mathcal{H}\left(\mathbf{x},\mathbf{p}\right)}{k_{B}T}\right\}d\mathbf{x}d\mathbf{p}. \tag{2.8}$$

As with the micro canonical ensemble, the integral is taken over all spatial $\mathbf{x}$ and momentum $\mathbf{p}$ components within the volume, and the $1/N!h^{3N}$ factor comes from particle indistinguishability and quantum mechanical considerations.

As this ensemble can exchange heat with its surroundings, via the heat bath, a system governed by it will come to equilibrium in the macrostate that minimises the Helmholtz free energy $A=U-TS$, with U begin the internal energy of the system and S being its entropy. As this free energy defines the equilibrium state of the system, it is extremely useful, however it is not directly accessible from simulations.

2.2.3 Isothermal-Isobaric Ensemble

As the name suggests, the isothermal-isobaric ensemble describes a system with a constant temperature (T) and pressure (P) with a fixed number of particles (N). This ensemble is often referred to as the NPT ensemble and couples the system to some external "piston" to allow it to expand and contract as well as a heat bath.

The partition function for this ensemble is given by

$$\Delta(N,P,T) = \frac{1}{V_0 N! h^{3N}} \int_0^\infty \iint \exp\left\{-\frac{\mathcal{H}(\mathbf{x},\mathbf{p}) + PV}{k_B T}\right\} d\mathbf{x} d\mathbf{p} dV, \tag{2.9}$$

here V_0 is a reference volume required to keep the partition function dimensionless, and the order of integration is important. We must integrate over the spatial and momentum components $\mathbf{x}$ and $\mathbf{p}$ before the volume component as the position component is restricted to the domain defined by the volume.

As with the canonical ensemble, a system governed by this ensemble will come to equilibrium in a microstate that minimises a free energy parameter, however under the isothermal-isobaric ensemble the free energy that will be minimised is the Gibbs free energy, $G = A + PV = U + PV - TS$ rather than the Helmholtz free energy A.

2.2.4 Isoenthalpic-Isobaric Ensemble

The isoenthalpic-isobaric ensemble describes a system with constant enthalpy $H = U + PV$, pressure P and number of particles N, often referred to as the NPH ensemble. The partition function for this ensemble is given by,

$$\Gamma(N,P,H) = \frac{1}{V_0 N! h^{3N}} \int_0^\infty \iint \delta\left(\mathcal{H}(\mathbf{x},\mathbf{p}) + PV - H\right) d\mathbf{x} d\mathbf{p} dV, \tag{2.10}$$

as with the isotherm-isobaric ensemble the order of integration is important. This partition function has a similar form to the microcanonical ensemble partition function Ω as where Ω describes a surface of constant energy, Γ describes a surface of constant enthalpy.

2.3 Molecular Dynamics

The statistical mechanical ensemble partition functions described in section 2.2 allow us to calculate macroscopic properties of a system purely by considering the Hamiltonian. However, solving these integrals analytically becomes impractical as the system complexity increases; this leaves us with the options of reducing complexity or using numerical methods to estimate the integrals.

Molecular dynamics (MD) is a broad term for numerical methods for calculating the time integral of the Hamiltonian under the constrains of a statistical mechanical ensemble. If the system obeys the ergodic hypothesis, given a long enough amount of time, all microstates are equally likely, then a long enough MD trajectory, or enough samples of stochastic trajectories, will allow us to determine the thermodynamic properties of the system [14].

2.3.1 Integrators

In order to generate these MD trajectories, we need methods to perform the time integration. Here we will discuss the integrators used for MD simulations in this work.

Velocity Verlet

For simulations at constant energy under the microcanonical ensemble (NVE) we can the use the velocity Verlet algorithm [14]. This algorithm takes Hamilton's equations,

$$\dot{q}_i = \frac{\partial \mathcal{H}}{\partial p_i} \qquad \dot{p}_i = -\frac{\partial \mathcal{H}}{\partial q_i}, \tag{2.11}$$

Taylor expanding them around a time step $t + \Delta t$. This discretises the time domain, and by ignoring terms of order 3 or higher in Δt we generate expressions for the incremental changes in position ($\mathbf{r}$) and velocity ($\mathbf{v}$),

$$\mathbf{r}_i(t + \Delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t - \Delta t) + \frac{\Delta t^2}{m_i}\mathbf{F}_i(t) \tag{2.12}$$

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i + \frac{\Delta t}{2m_i}\left[\mathbf{F}_i(t) + \mathbf{F}_i(t + \Delta t)\right]. \tag{2.13}$$

The velocity component requires two evaluations of the force $\mathbf{F}$, one at time t and one at time $t + \Delta t$. If $\mathbf{F}$ is independent of $\mathbf{v}$, then we can simply evaluate the

forces before and after moving the particles to determine $\mathbf{v}$. However if $\mathbf{F}$ has some dependence on $\mathbf{v}$ then a more involved integration scheme may be required. As there is no external energy source, this integration scheme will keep the system at constant energy, and the simulation box parameters will remain constant giving the requirements for the NVE ensemble, assuming we do not allow particles to leave the system.

Nosé-Hoover

The velocity Verlet scheme described above allows the particle motions to be calculated, but does not allow us to control any of the system's thermodynamic properties such as temperature and pressure required by the NVT, NPH and NPT ensembles. To allow control of these variables, Nosé [15] suggested the addition of an additional variable, and its associated conjugate momenta, to the Hamiltonian. This additional variable couples the Hamiltonian to some constant temperature T via a damping coefficient. The form of these modifications was later modified by Hoover [16] and the concept of a Nosé-Hoover chain was introduced by Martyna et al [17]. A Nosé-Hoover chain extends the single additional variable to a chain of variables which each ensure that alterations to the previous variable are correctly drawn from the Maxwell-Boltzmann distribution. A complete discussion of the formalism and implementation of this is beyond the scope of this work.

These Nosé-Hoover chains couple the particle momenta to some external heat source, giving us a thermostat to control the system temperature. This allows us to sample the canonical (NVT) ensemble. We can modify the Hamiltonian in a similar way to couple the system to some extremal pressure and allowing the volume to fluctuate, this gives us a barostat to regulate the system pressure. Again, a full derivation an discussion of this is beyond the scope of this work. By combining the thermostat and barostat we can now sample all of the ensembles described in section 2.2, applying only the thermostat gives NVT, only the barostat gives NPH, and applying both the thermostat and the barostat give the NPT ensemble. These couplings generate the correct equilibrium distribution, however the mechanism is unphysical so simulations will not give us the same microscopic kinetics as a system surrounded by a much larger heat bath.

Canonical Sampling Thermostat

Nosé-Hoover is just one option for thermostating a MD simulation. Another approach to applying a thermostat are velocity rescaling methods. These methods simply evolve the MD simulation using velocity Verlet, or some other integrator, and rescale the velocities to recover the desired system temperature. In this work we will use the canonical sampling velocity rescaling thermostat described by Bussi et al. in [18]. This thermostat allows the particles to evolve in time, then uses a stochastic process, which we will not discuss here, to rescale the velocities to the Maxwell-Boltzmann distribution at the thermostat temperature. We can apply this thermostat with just the Verlet integration to achieve an NVT ensemble, or with the Nosé-Hoover barostat to achieve a NPT ensemble. As with Nosé-Hoover, this thermostat generates the correct equilibrium distribution to allow us to sample our ensemble, but does it via an unphysical mechanism.

2.3.2 Lennard-Jones Particles

So we have discussed methods of integrating the Hamiltonian to simulate the dynamics of particles, but we have yet to discuss how those particles interact with each other. For particles to interact with each other, we need to define an interaction potential. As this potential defines how the MD particles effect each other, it is a choice of what particles we are simulating. For each type of particle we wish to simulate, we can define a potential for how they interact with other particles of the same type, as well as particles of different type. For simplicity, and because it is all we require for this work, we will only discuss systems with a single type of particle, therefore requiring only one interaction potential.

One simple choice of interaction potential is the Lennard-Jones (LJ) potential, which is often used to simulate argon. This pairwise potential,

$$V_{\mathrm{LJ}}(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right], \tag{2.14}$$

is defined only in terms of the distance between the two particles r and two scale parameters, a length scale σ and an energy scale ϵ. The LJ potential, shown in figure 2.1, is slightly attractive at long range and repulsive at short range. The energy and length scales allow us to define a set of dimensionless units, table 2.1, to describe the properties of LJ systems.

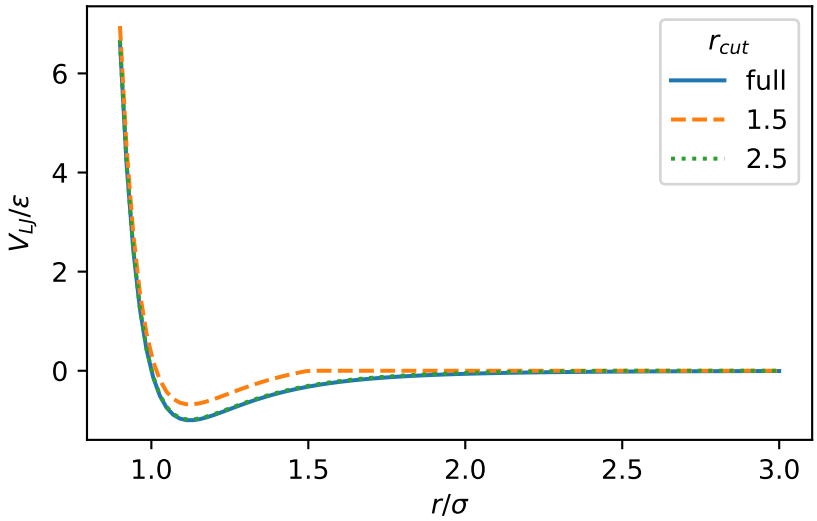


Figure 2.1: Comparison of cut off values r_{cut} for the truncated and shifted Lennard-Jones potential to the full potential.

Property	Dimensionless form
Length	$r^* = r\frac{1}{\sigma}$
Energy	$E^* = E\frac{1}{\epsilon}$
Temperature	$T^* = T\frac{k_B}{\epsilon}$
Time	$t^* = t\sqrt{\frac{\epsilon}{m\sigma^2}}$
Pressure	$P^* = P\frac{\sigma^3}{\epsilon}$
Density	$\rho^* = \rho\sigma^3$
Force	$F^* = F\frac{\sigma}{\epsilon}$

Table 2.1: Dimensionless forms of properties for the Lennard-Jones system, here m is the particle mass, usually set as $m = 1$.

The Lennard-Jones potential, equation 2.14, has an infinite range. This poses a problem for computational simulations, as we cannot have a infinite simulation box, and if we have periodic boundary conditions, each particle will interact with itself an infinite number of times, so a loop over interacting pairs would take infinitely long to evaluate. To reduce this to something we can realistically compute, we define a cut off point r_{cut} at which we truncate the potential to allow us to compute interactions over a finite range. At this cut off point $V_{\text{LJ}}(r_{\text{cut}}) \neq 0$ so there will be some discontinuous jump in force at r_{cut}. To remove this jump we can also shift the potential so that the truncated and shifted potential $V_{\text{LJTS}}(r_{\text{cut}}) = 0$. This shift gives the potential

$$V_{\text{LJTS}}(r) = \begin{cases} V_{\text{LJ}}(r) - V_{\text{LJ}}(r_{\text{cut}}) & \text{if } r < r_{\text{cut}} \\ 0 & \text{otherwise.} \end{cases} \tag{2.15}$$

A comparison of the unmodified and truncated shifted LJ potentials are shown in figure 2.1. The difference in these potentials for different values of r_{cut} can have an effect on the physical properties of the system such as the solid and liquid densities.

Modified Lennard-Jones Potential

As discussed above, the truncated and shifted LJ potential changes the minimum energy of the potential leading to potentially large effect on the system's properties. In [19], Broughton and Gilmer propose a different truncation method which uses two radii of interest r_1 and r_2 and aims to closely match the unmodified LJ potential at $r < r_1$ and smoothly approach 0 over the range (r_1, r_2). They choose $r_1 = 2.3\sigma$, $r_2 = 2.5\sigma$ and define their modified LJ potential as

$$V_{\text{LJM}}(r) = \begin{cases} 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right] - C_1 & r \leq r_1 \\ C_2 \left(\frac{\sigma}{r}\right)^{12} + C_3 \left(\frac{\sigma}{r}\right)^{6} + C_4 \left(\frac{r}{\sigma}\right)^{2} + C_5 & r_1 < r < r_2 \\ 0 & r \geq r_2 \end{cases} \tag{2.16}$$

with constants $C_1 = 0.016132\epsilon$, $C_2 = 3136.6\epsilon$, $C_3 = -68.069\epsilon$, $C_4 = 0.08331\epsilon$, and $C_5 = 0.74689\epsilon$. This form gives a minimum energy of $\min(V_{\text{LJM}}) = -0.983868\epsilon$ which is much closer to the minimum energy of the full LJ potential, $\min(V_{\text{LJ}}) = -\epsilon$, than the truncated and shifted potential with a cut off of $r_{\text{cut}} = 2.5\sigma$, $\min(V_{\text{LJTS}}|r_{\text{cut}} = 2.5\sigma) = -0.93\epsilon$.

2.4 Nucleation

Nucleation is a stochastic process in which a more stable phase **B** forms from some metastable phase **A** [20]. During this process the system goes through some critical point, also call a transition state, at which the system is equally likely to continue to state **B** or return to state **A**. In the absence of some influence, spontaneous nucleation can take a long time to occur and so can reasonably be categorised as a rare event.

2.4.1 Rare Event Statistics

As discussed, the long timescales associated with spontaneous nucleation categorises it as a rare event. In order to introduce the statistics associated with these events, we will consider a simple model. The model we shall consider is the transition between two states of a particle undergoing Brownian motion in a 1D double well potential, figure 2.2. In this example, a particle starts at some position in well **A** and moves on the potential nudged by some random forces. In the case of Brownian motion these forces are proportional to the temperature.

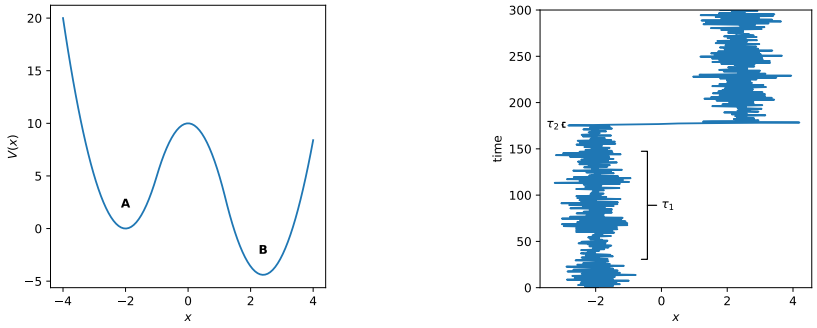
To begin with, the particle just explores well **A** and forms a quasi-stationary distribution which is Boltzman like, provided that the lifetime of the metastable state is long enough, which is dependent on the temperature and barrier height between the wells. Given enough time, the random forces will push the particle over the barrier. Figure 2.2 shows an example trajectory for a particle in this system.

From this trajectory we see that it takes a long time before the particle crosses the barrier, τ_1, but the crossing itself takes comparatively very little time τ_2. If we repeat the experiment, we can estimate the lifetime of state **A** by averaging the time taken before a particle first transitions from **A** to **B**, giving the mean first passage time (MFPT) $\tau_{\mathrm{MFP}} = \langle \tau_1 + \tau_2 \rangle$.

As the particle thermalises and forms a quasi-stationary distribution in the well before transitioning, we can classify the transitions as discrete events, and therefore model the transitions using Poisson statistics. The Poisson distribution gives the probability of some discrete random variable X with expectation value λ being some integer x [21],

$$P(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}. \tag{2.17}$$

If we define a barrier crossing rate $k_{\mathrm{MFP}} = 1/\tau_{\mathrm{MFP}}$, then the expected number of



(a) Potential. (b) Brownian motion.

Figure 2.2: Double well potential V (a) and Brownian motion on V (b) at $T = 2$. With $V(x) = 5(x+2)^2$ for $x \leq -1$, $10 - 5x^2$ for $-1 < x \leq 1.2$, and $5(x-2.4)^2 - 4.4$ for $x > 1.2$

barrier crossings in time t will be $k_{\mathrm{MFP}}t$. Using the Poisson distribution, equation 2.17, we can determine the probability of observing N events in time t,

$$P(N|t) = \frac{(k_{\mathrm{MFP}}t)^N e^{-k_{\mathrm{MFP}}t}}{N!} \tag{2.18}$$

2.4.2 Classical Nucleation Theory

Classical nucleation theory (CNT) treats the growth of a cluster of particles as a Markovian random walk on a 1D free energy surface. This free energy surface is defined as the sum of two competing terms, an energy benefit from adding particles to the cluster (increasing the volume), and a penalty for increasing the interfacial surface area,

$$F(n) = -|\Delta\mu|\, n + \phi\gamma n^{\frac{2}{3}} \tag{2.19}$$

where n is the number of particles in the cluster, $|\Delta\mu|$ is the chemical potential between solid and liquid phases, γ is the interfacial free energy, and ϕ is a shape factor. If $\phi = 4\pi(3/4\pi)^{\frac{2}{3}}$ then $\phi n^{\frac{2}{3}}$ will be the surface area of a spherical cluster of n particles. This free energy surface forms a barrier to nucleation, shown in figure 2.3, as there is some potential which must be overcome before it is energetically favourable for the cluster to continue growing. This is similar to the double well potential from section 2.4.1, and depending on the size of the barrier means that nucleation can be modeled as a rare event.

The top of the free energy barrier represents the transition state, or critical

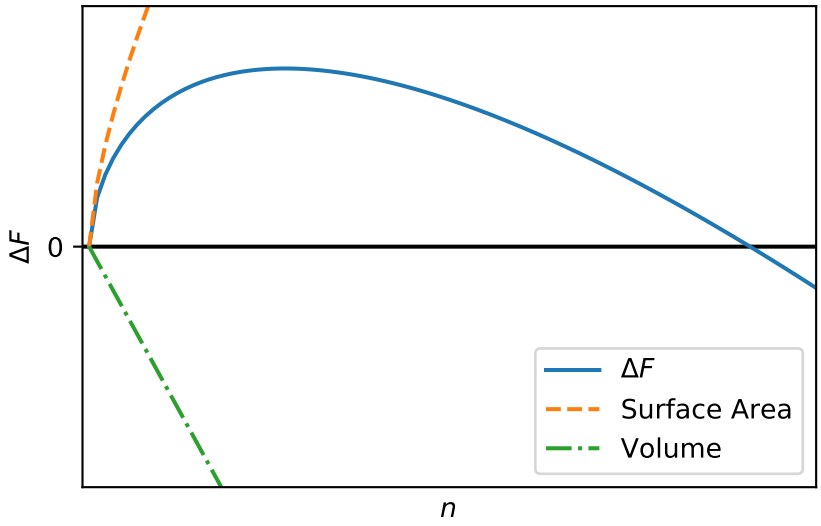


Figure 2.3: Form of the free energy landscape assumed in classical nucleation theory ΔF with contributions from the volume (Green) and surface (Orange) terms shown for illustration.

nucleus size n_c. This is the point at which the cluster is equally likely to grow and shrink. By solving $\partial F(n)/\partial n = 0$ we can determine this critical cluster size as,

$$n_c = \left(\frac{3}{2}\frac{\phi\gamma}{|\Delta\mu|}\right)^3. \tag{2.20}$$

By substituting 2.20 into 2.19 we can get the height of the free energy barrier,

$$F_c = F(n_c) = \frac{n_c\,|\Delta\mu|}{2}. \tag{2.21}$$

From Boltzmann statistics, the probability of transitioning from a state with energy E_1 to a second state with energy E_2 is given by the probability $P\left(E_2|E_1\right) = \exp\left\{\frac{E_1-E_2}{k_BT}\right\}$, therefore the probability of a cluster being at the top of the barrier is

$$P\left(n_c\right) = \exp\left\{\frac{-F_c}{k_BT}\right\}. \tag{2.22}$$

CNT makes the assumption that only one particle is added to, or removed from the nucleus at a time as it models growth as a random walk. The rate of addition / removal events occurring is given by the diffusion coefficient, D. The

product of D with the density of particles, ρ, gives the rate at which the size of the cluster will change. If we are interested in the nucleation rate, i.e. the rate at which the cluster grows beyond the critical nucleus size and crystallises, then we need to know the ratio of attachments to detachments at the critical nuclear size. This ratio is given by the Zeldovich factor,

$$Z = \sqrt{-\frac{1}{2\pi k_B T} \left. \frac{\partial^2 F(n)}{\partial n^2} \right|_{n=n_c}}. \tag{2.23}$$

Given all of these pieces we can express the nucleation rate as,

$$J = \rho Z D \exp\left\{\frac{-F_c}{k_B T}\right\}. \tag{2.24}$$

We can use this nucleation rate to determine how many nucleation events we are likely to see in a given time frame using the Poisson distribution discussed in section 2.4.1.

2.5 Reaction Coordinates

When talking about nucleation, we used the position of the particle to determine how far the particle was from either of the wells, and to track its progress. The tracking of progress from the reactant state **A** to the product state **B** is useful, not only to see when the system has made the transition, but is an integral part of of some computational methods discussed later. In order to track the progress of a system we typically reduce all of the simulation dimensions, 3 position and velocity for each particle plus any thermostat and barostat coordinates, into one scalar collective variable referred to as the reaction coordinate.

To be effective, this reaction coordinates should take different values when the system is in states **A** and **B**, and transition smoothly as the system moves between them. In the simple system from section 2.4.1 the position coordinate already functions as good reaction coordinates, so no additional calculations are required.

In other systems, the choice of reaction coordinates may not be as simple, and as system complexity increases, combinations of reaction coordinates may be required to fully describe the free energy landscape of the problem.

2.5.1 Cluster Size

When concerned with nucleation and crystal growth, a physically motivated choice of reaction coordinate is the cluster size. This allows simulations to be linked to both experiments, where number and size of clusters is observable, and with classical nucleation theory which will become useful later, section 2.6.3 and chapter 4.

While counting the number of atoms in a cluster is simple in concept, we must first determine which atoms are "crystal" and which are "liquid". One way to do this is to look at the local order surrounds each particle, as "crystal" particles should have more ordered surroundings than "liquid" particles.

Steinhardt Parameters

One method of calculation how ordered a region is around a particle is the Steinhardt parameter [22]. These parameters come in two forms, a scalar Q_l and a vector q_{lm} given by

$$q_{lm} = \frac{4\pi}{2l+1}\bar{Y}_{lm}, \tag{2.25}$$

and

$$Q_l = \sqrt{\sum_{m=-l}^{m=l} q_{lm} q_{lm}^*} \tag{2.26}$$

where $\bar{Y}_{lm}$ is a combination of spherical harmonics,

$$\bar{Y}_{lm} = \frac{1}{N}\sum_{j=1}^{N} Y_{lm}\left(\mathbf{r}_{ij}\right), \tag{2.27}$$

in which the sum is taken over the particle's neighbours. When choosing which particles to consider as neighbours, it is typical to use all the particles within some cut-off. The cut-off can be chosen based on the interaction potential of the particles.

The value of l can be chosen based on the expected crystal geometry, if it is known, or based on computational limits. We see from equation 2.25 that as l is increased, higher order spherical harmonics are used. This also means the size of the vector $\mathbf{q}_l$ increases with l, its dimensionality being $2l+1$.

The vector $\mathbf{q}_l$ represents the direction of local order around the particle i, if we take the dot product of the $\mathbf{q}_l$ vectors for two particles, i and j, then we get a scalar measure of how similar the local surroundings of the particles are.

The more neighbours a particle has with a similarly ordered surrounding, the more likely it is to be solid [23]. We define a coordination number for each particle in the system as the number of neighbours it has where the dot product of their vectors is greater than some threshold. Then, based on the coordination numbers we can decide which atoms to label as solid. This labelling again requires choosing a threshold value, ideally based on the topology of the crystal structure. This method of determining cluster sizes, introduced by ten Wolde in [23], is commonly used with $l = 6$ for cluster calculations in Lennard-Jones systems.

Largest cluster

In an MD simulation it could be the case that not all of the particles labelled as solid are in the same crystal structure, in fact it is almost never the case. If we want to describe the system in terms of the size of a crystal, it often makes sense to choose the largest structure. To determine which is largest and how many particles are included in it we have to decide which particles are in which crystal. This is usually done by defining connections between solid particles, either by considering the $\mathbf{q}_l$ dot product or just by looking for other solid particles within some cut off. Once connections have been defined, a clustering algorithm can be used to find the size of the largest cluster, which in this case corresponds to the size of our largest crystal.

2.5.2 The Committor

The size of largest cluster is a physically motivated reaction coordinate, but many more options exist. In section 3.1 we will discuss some other options, but here we will introduce the ideal reaction coordinate, the committor [24, 25]. For any transition from a state **A** to **B**, for every microstate of the system there is a probability, p_B, that it will evolve to state **B**, before returning to state **A**, this probability is the committor. This is the ideal choice of reaction coordinate as by definition it will increase as the system gets closer to state **B**, and all states with the same likelihood of reaching **B** will have the same value. p_B also has the additional benefit that any transition states, which may be hard to identify with other reaction coordinates, will have a value of $p_B = \frac{1}{2}$. While p_B is an ideal reaction coordinate choice in theory, in reality it can be computationally expensive to calculate, hence other reaction coordinates, like the size of largest cluster are often used instead.

2.6 The Timescale Problem

If nucleation is correctly modelled as a rare event, then events only happen with a low probability, meaning that the metastable state or phase has a long lifetime. MD simulations model the atomistic behaviour of the particles within a system which, by their nature, evolve on a short timescale which means that due to computational limitations, we are only able to run MD simulations on small systems for short timescales. This contradiction in timescales is the timescale problem, and means that, except for systems with very simple dynamics or very low free energy barriers, it is often impractical to take simply run a simulation until nucleation is observed, as we did for the barrier crossing in section 2.4.1.

There are two paths to combat this problem, make the MD simulation faster, or find some way to avoid simulating the system for the large period of time that nothing of interest is happening. The MD simulations can be made faster by using parallel algorithms or newer hardware, however that speedup does have practical limits. We could also use simple models, however that again has practical limits as we may lose the physics we're trying to simulate. If we accept that the simulations are practically as good as they are going to get, then we are left with the option of trying to limit our simulation to areas of interest. Here we will briefly introduce some methods to combat this timescale problem.

2.6.1 Free Energy Methods

Free energy methods in general use a biasing potential to encourage the system to explore more of the free energy landscape. They can then approximate the free energy of the system using the applied bias. These methods give access to the energy barrier, but not necessarily the nucleation mechanism.

One such free energy method is metadynamics [26]. In this method the system runs for a small number of time steps, then a Gaussian bias is applied to the potential. This process is repeated until a threshold reaction coordinate is reached and the biased potential is flat. Once the biased potential is flat, it can be inverted to recover the free energy landscape of the system.

Another free energy method is umbrella sampling [27]. In umbrella sampling the system is retained within a harmonic well to force it to explore the free energy in a narrow region. This well is then moved along the reaction coordinate until the desired region of the free energy landscape has been explored. After the desired

range of the reaction coordinate has been explored, the results for each sampling region can be combined to recover the underlying free energy landscape.

2.6.2 Path Sampling

Path sampling methods attempt to selectively sample a system to minimise the time spent in the starting state **A** and ending state **B**, and maximise the time spent simulating regions around the transition state or on transition paths between the two state. For these methods we need some reaction coordinate to determine if we are in either the starting or ending state and track progress between them.

Transition Path Sampling

Transition path sampling (TPS) [28] is a path sampling method which stochastically samples the ensemble of transition paths connecting states **A** and **B**. In TPS, we select a point along a transition path and modify it to generate new paths. The method of modification can vary between implementations; here we will briefly discuss two, shooting and shifting. If we are using multiple modification types, we choose which to apply randomly based on some predefined probabilities.

For shooting moves we select a random time slice t_0, sampling uniformly in the range $[0, t_{\text{max}}]$, from our starting path x_{current}. We then perturb the momenta in the microstate $x_{\text{current}}(t_0)$, the method of perturbation should be compatible with the desired thermodynamic ensemble. After perturbing the momenta we propagate the microstate forwards in time to $t = t_{\text{max}}$ and backwards in time to $t = 0$ to generate a new path x_{new}.

For shifting moves we choose some time shift δt drawn from a Gaussian distribution, with mean zero and some pre-decided variance. If $\delta t > 0$ we propagate the path forwards in time from $x_{\text{current}}(t_{\text{max}}) \rightarrow x_{\text{current}}(t_{\text{max}} + \delta t)$. However if $\delta t < 0$ we propagate the path backwards in time $x_{\text{current}}(0) \rightarrow x_{\text{current}}(0 - \delta t)$. We then remove the added time δt from the end of the trajectory that was not propagated from and shift our time domain by $\pm\delta t$ so the path starts at time $t = 0$ and ends a t_{max} with the positions and momenta shifted in time. The time shifted path is now our new path x_{new}.

If the starting point of the new path is still in the starting state $x_{\text{new}}(0) \in \mathbf{A}$, and the end of the new path is still in the ending state $x_{\text{new}}(t_{\text{max}}) \in \mathbf{B}$ then the new path is accepted. Once a path is accepted it is added to the ensemble

and either saved or relevant parameters are calculated from it then it replaces the current path x_{current}. Once a path has been accepted or rejected the process of selecting a modification type and modifying the current path is repeated until some ending criteria is met, this criteria could be some number of sampled trajectories, the required sampling of some parameter, or a computational time limit. TPS is a widely used method of generating transition paths however it does require a starting path to modify and, depending on the modification processes used, it can struggle to find transition paths which take a significantly different route than the starting path [29].

Forward Flux Sampling

Another variety of path sampling is forward flux sampling (FFS) [30]. In FFS we mark out interfaces in a chosen reaction coordinate, λ, to act as checkpoints, λ_i. To begin we run from some starting point in **A** and record the system state every time our reaction coordinate crossed the interface λ_0 in the forwards direction. These states form both the starting points for the FFS calculation and give us the flux out of **A**, Φ_A.

We now randomly choose one of our starting points and launch a trajectory from it. The trajectory is run until it either reaches the next interface, λ_1 in this case, or returns to the basin **A**. We do this with multiple trajectories which allows us to calculate the flux from λ_0 to λ_1, $\Phi_{0,1}$.

This process is repeated, each time running until the trajectory reaches λ_i or returns to **A**, until we reach the final interface λ_N. Once **B** has been reached by crossing λ_N the probability of going from λ_0 to λ_N can be calculated as

$$P_{A,B} = \prod_{0}^{N-1} \Phi_{i,i+1}. \tag{2.28}$$

Meaning the flux from **A** to **B** is given by

$$\Phi_{A,B} = \Phi_A \prod_{0}^{N-1} \Phi_{i,i+1}, \tag{2.29}$$

which in the context of nucleation is our nucleation rate.

As well as the two path sampling methods discussed there are many other variants including some methods which allow for time evolution of the free energy

such as Non-Stationary Forward Flux Sampling [31] and Stochastic Process Rare Event Sampling [32].

2.6.3 Seeding Methods

The last approach for avoiding the timescale problem we shall discuss, and the one which forms the basis of this work is seeding methods. These methods get around the timescale problem by generating systems with preformed seed nuclei. These seeded systems are simulated and the trajectories used to parametrise the CNT equations from section 2.4.2. These methods allow us to calculate the nucleation rate as well as the free energy barrier height and, depending on the particular method use, an estimation of the free energy. As these methods are based on CNT, they require that the nucleation pathway is well described by the cluster size n. For this to be the case, all of the other degrees of freedom in the underlying simulation must be in quasi equilibrium, that is they form an equilibrium-like distribution at each cluster size n. If this is not the case, then there is effectively some parameter other than n influencing the progression towards nucleation, and crystallisation.

Here we will introduce two variants of seeding method, which we will refer to as the single seed [2] and single temperature [33] methods, but will save the complete implementation details and discussion for chapter 4.

Single Seed Method

In this method, we take a single preformed and thermalised seed and then find the temperature at which this seed is at the critical nucleus size, n_c [2]. To do this we take the seed and run it forwards at several temperatures. Ideally we want a selection of temperatures close to and evenly distributed around the critical temperature, T_c (This will be guesswork as until running the simulation we have no idea how close to T_c we are). This will give us a selection of trajectories, in some of which a nucleus is growing, and shrinking in others. If we fit these trajectories, we can infer the temperature at which the seed would neither grow nor shrink (on average), which would be T_c.

Now we have T_c and also n_c. By running trajectories at T_c we can calculate the diffusion coefficient D and by running bulk simulations with no seed we can calculate the density of the fluid phase. This gives us all the required information to calculate the nucleation rate using equation 2.24.

Single Temperature Method

In this method, we take several seeds and run them at one temperature [33]. We want some pre-critical and some post-critical seeds. As with the single seed method, we get some shrinking and some growing trajectories and we can fit them to infer the critical nucleus size n_c. These trajectories are also used to calculated the gradient of the free energy landscape $\frac{\partial F}{\partial n}$ which we can fit to the differential of equation 2.19. From this we can calculate the nucleation rate as in the single seed method using equation 2.24.

Chapter 3

Relevant Literature

In the previous chapter (sections 2.4.2 and 2.6.3) we introduced CNT and two seeding methods based on it along with the main assumptions we intend to explore in chapters 4 and 5, those being that n is a slow degree of freedom and that the trajectories of nuclei size are well described by Markovian random walks on the CNT free energy landscape. We also introduced reaction coordinates and described that while n is a physically motivated choice it may not be the ideal candidate for all simulations. We also stated that the ideal reaction coordinate for any transition is the committor p_B which we will attempt to predict from simpler metrics and the current microstate for magnetisation reversal in the 2D Ising model in chapter 6.

In this chapter, we will give brief overviews of some other works which take different approaches to the same problems, or similar approaches applied to different problems. We will look at comparison of reaction coordinates indicating a better reaction coordinate than n, motivating our models in chapter 6. This analysis also uses a maximum likelihood approach similar to the one we will employ for memory identification in chapter 5. We will also look at an exploration of how well nucleation is described by a Markovian random walk for magnetisation reversal in the 2D Ising models, which takes a different approach to the one we employ in chapter 5. Finally we consider a study comparing CNT and FFS result for magnetisation reversal in the 2D Ising model which, as well as providing an accurate free energy which we will use to generate our reference stochastic processes in chapter 5, sets an expectation for the memory within that model, as well as the validity of using n as a reaction coordinate.

3.1 Reaction Coordinate Selection

As discussed in section 2.5.1, the Steinhardt parameters allow us to compute the similarity in the local order around two particles. In [23], ten Wolde et al use these parameters to distinguish between liquid, and crystalline phases, body centre cubic (bcc) and face centre cubic (fcc), of a Lennard-Jones system in the same manner discussed in section 2.5.1. In this work, ten Wolde proposes the size of the largest cluster, n, as an appropriate reaction coordinate for crystallisation of the Lennard-Jones fluid from the melt. This reaction coordinate is determined by atoms in a particular region of the system, the region with the largest cluster. Ten Wolde also shows that if we take a purely global reaction coordinate, such as $Q_6^{\text{global}} = 1/N_{\text{particles}} \sum_i^{N_{\text{particles}}} Q_{6,i}$ with $N_{\text{particles}}$ being the number of particles in the system as done in [23], then the pre-critical region is dominated by lots of small clusters which combine as we approach the top of the free energy barrier. This is because there is an entropic benefit from having many small clusters rather than a single larger one. The importance of this benefit relative to the interfacial free energy penalty of having many clusters is artificially enhanced in a finite size system. Biasing a local order parameter to promote nucleation circumvents this by encouraging only a single cluster to form.

While the size of largest cluster is a physically motivated reaction coordinate, and is important for applying CNT (section 2.4.2), there are many other possible nucleus size metrics that can be used as the reaction coordinate for nucleation from the melt. In [1], Beckham et al. examine several reaction coordinates to determine which of them provides the most accurate description of the liquid-solid phase transition for Lennard-Jones fluids, and potentially other spherically symmetric fluids. The order parameters tested include the size of largest cluster [23] which is purely based on the number of particles in the cluster n; Q_6^{global} which is entirely dependant on the global structure; the ratio of maximum and minimum moments of inertia for the largest cluster, $I_{\text{max}}/I_{\text{min}}$, which is dependant on the cluster shape; the average coordination number of the particles in the cluster $\langle c_{\text{cluster}} \rangle$; Q_6^{cluster}, the average Q_6 of all the particles in the cluster; and $n \cdot Q_6^{\text{cluster}}$ which combines the size of the cluster with a measure of its local order.

In order to compare these reaction coordinates, Beckham et al. use the likelihood maximisation method described in [24]. This method uses aimless shooting, a modification to TPS described in section 2.6.2 which draws new momenta from the

appropriate distribution at each shooting step rather than making small perturbations to the current momenta. The aimless shooting algorithm is used to generate a ensemble of transition paths which can then be used to calculate the likelihood that those paths are well described by a selection of reaction coordinates.

For any microstate $\mathbf{x}$, the probability that is is on a transition path is $p(\mathrm{TP}|\mathbf{x})$, this probability is approximated in aimless shooting by taking the ratio of accepted shooting moves N_{Accept} to attempted N_{Attempt} shooting moves for the microstate $\mathbf{x}$,

$$p(\mathrm{TR}|\mathbf{x}) = \left.\frac{N_{\mathrm{Accept}}}{N_{\mathrm{Attempt}}}\right|_{\mathbf{x}}. \tag{3.1}$$

If we have a good reaction coordinate $R(\mathbf{x})$, then this probability should only be dependent on the reaction coordinate $p(\mathrm{TP}|\mathbf{x}) = p(\mathrm{TP}|R(\mathbf{x}))$. Likelihood analysis also uses a model function for $p(\mathrm{TP}|R)$ which has a maximum at the transition state and decays to 0 on both sides of the peak. In [24], Peters and Trout choose

$$p(\mathrm{TP}|R) = p_0\left[1 - \tanh\left(R\right)^2\right] \tag{3.2}$$

as the model function where p_0 is a constant, which for diffusive dynamics gives the committor probability as [24, 34],

$$p_B(R) = \frac{1}{2}\left[1 - \tanh\left(R\right)^2\right]. \tag{3.3}$$

This model function allows us to calculate the likelihood that a given reaction coordinate $R(\mathbf{x})$ is representative of the ensemble,

$$L(R) = \prod_{i}^{B} p_B\left(R\left(\mathbf{x}_i\right)\right) \prod_{i=1}^{\bar{B}} \left[1 - p_B\left(R\left(\mathbf{x}_i\right)\right)\right], \tag{3.4}$$

where B is the number of paths that end in the final state $\mathbf{B}$ and $\bar{B}$ is the number of paths which do not for a microstate $\mathbf{x}_i$. Using this methodology, Beckham and Peters calculate the likelihoods for a wide range of reaction coordinates including those mentioned previously, the results of which are shown in table 3.1. These results show that while n is by no means the worst choice of reaction coordinate, the transition path ensemble is better represented by $n \cdot Q_6^{\mathrm{cluster}}$ which includes local structural information as well as the cluster size. This result is echoed by similar work from Moroni et al. [35] in which they use a variety of path sampling algorithms

Reaction Coordinate	Likelihood
n	$\exp(-1726.6)$
Q_6^{cluster}	$\exp(-2244.0)$
$n \cdot Q_6^{\text{cluster}}$	$\exp(-1570.8)$
$\langle c_{\text{cluster}} \rangle$	$\exp(-1860.6)$
Q_6^{global}	$\exp(-1909.3)$
$I_{\text{max}}/I_{\text{min}}$	$\exp(-2175.2)$

Table 3.1: Likelihood scores for a selection of reaction coordinated from [1].

to study nucleation in Lennard-Jones systems and find that nucleation proceeds by a number of different pathways concluding that a accurate description of nucleation requires consideration of the size, structure, and shape of a forming nucleus.

In chapter 5 we will use a related likelihood approach to test whether the dynamics of a reaction coordinate are well described by a candidate model stochastic process (with and without memory) and hence compare those processes in terms of their ability to capture those dynamics.

3.2 Testing the Markovanitiy of the Ising model

In [36], Kuipers and Barkema use the 2D Ising model to examine how well the nucleation process can be described as Markovian. To do this they take the Fokker-Plank equation, discretise it and compare it to the transition matrices generated during magnetisation reversal in the 2D Ising model with both spin flip and spin exchange dynamics.

For a Markovian process, the probability of having a cluster of size n at time $t + \delta t$, $p(n, t + \delta t)$ is only dependent on $p(n + \delta n, t)$ for small changes in cluster size δn. The time evolution of this process can be described by the Fokker plank equation [20],

$$\frac{\partial}{\partial t} p\left(n,t\right) = \frac{\partial}{\partial n}\left[v\left(n\right) p\left(n,t\right)\right] \frac{1}{2} \frac{\partial^2}{\partial n^2}\left[D(n) p(n,t)\right], \tag{3.5}$$

where $v(n)$ is the drift velocity, and $D(n)$ is the diffusion coefficient. By discretising

into cluster sizes bins with width Δn and time steps Δt we get

$$p_i(t+\Delta t) = \sum_j \left(\delta_{ij} + \Delta t M_{ij}\right) p_j(t) \tag{3.6}$$

as the evolution of the probability of having cluster sizes in bin i. Where δ_{ij} is the delta function and M_{ij} is a tridiagonal matrix given by

$$\begin{aligned} M_{i-1,i} &= \frac{v_i}{2\Delta n} + \frac{D_i}{2\Delta n^2}, \\ M_{i,i} &= -\frac{D_i}{\Delta n^2}, \\ M_{i+1,i} &= -\frac{v_i}{2\Delta n} + \frac{D_i}{2\Delta n^2}. \end{aligned} \tag{3.7}$$

This means that the probability distribution of cluster size at an time $t+\tau$ can be expressed as

$$p(t+\tau) = \exp\left(\tau M\right) p\left(t\right), \tag{3.8}$$

for any time interval τ.

To compare this Fokker-Planck description to the spin flip and spin exchange Ising model dynamics, we need to generate a transition matrix for each set of dynamics. This is done by descretising the cluster size into bins and recoding the transition between bins. These transition can then be used to generate the transition matrix

$$T_{ij} = \frac{N_{ij}}{\sum_{i'j} N_{i'j}}, \tag{3.9}$$

where N_{ij} is the number of transitions from bin i to bin j. As T is constructed from counts we can define a standard error for it as,

$$\epsilon_{ij} = \frac{\sqrt{N_{ij}}}{\sum_{i'} N_{i'j}}. \tag{3.10}$$

Comparing these two models means that the Fokker-Planck equation that best describes the transition matrix T is given by minimising

$$\sum_{i,j} \left(\frac{T_{ij} - \exp\left(\tau M_{ij}\right)}{\epsilon_{ij}} \right)^2. \tag{3.11}$$

Using this analysis approach, Kuipers and Barkema [36] find that for long enough intervals, spin flip dynamics exbibits Markovian behaviour, although this does breakdown at small time intervals and small cluster sizes. The breakdown at small cluster sizes is attributed to the assumption of spherical clusters being inaccurate for these cluster sizes. As for spin exchange dynamics, Kuipers and Barkema find that they do not behave in a Markovian manner and suggest that other factors should be considered when desiring nucleation, much like the reaction coordinate discussion from section 3.1.

In chapter 5 we will use a related but different technique to attempt to quantify the strength of any non-Markovian behaviour by comparing Ising model simulation under magnetisation reversal to reference stochastic processes.

3.3 Ising Model Nucleation and Free Energy

In [37], Ryu and Cai use both CNT and forward flux sampling (FFS) to calculate the nucleation rate for the 2D and 3D Ising model with spin flip dynamics. They compare the results of these two methods to determine if this model is well described by CNT.

For the CNT calculations, Ryu and Cai use a version of the CNT free energy, equation 2.19, with some extra correction terms which are fitted from umbrella sampling results. For the Ising model, the chemical potential $\Delta\mu$ is simply twice the external field, $\Delta\mu = 2h$. The interfacial free energy is slightly harder to calculate, with the interfacial surface free energy being given by

$$F_\gamma(n) = 2\sqrt{\pi n}\gamma_{\text{eff}}(T). \tag{3.12}$$

The analytic form of γ_{eff} for the 2D Ising model is given in [38]. The two corrective terms added to the CNT free energy are a logarithmic term and constant term [37] giving the complete free energy as

$$F(n) = -2h + 2\sqrt{\pi n}\gamma_{\text{eff}}(T) + \tau k_B T \ln(n) + d(T), \tag{3.13}$$

Where τ and $d(T)$ are correction parameters which will be fitted along with γ_{eff} from the umbrella sampling results.

For the FFS, described in section 2.6.2, Ryu and Cai use the size of largest cluster as the reaction coordinate. Comparing the nucleation rates from CNT and

FFS, Ryu and Cai find them to be in agreement for a wide range of nucleation rates for both the 2D and 3D Ising model, with spin flip dynamics. As the nucleation rates are similar, they conclude that the assumptions made in CNT are valid for these models.

We will use the accurate free energy from this in chapter 5 when constructing the reference stochastic processes for examining memory in Ising model seeded trajectories. As the CNT and FFS results are in good agreement we expect that the memoryless dynamics to be a good description of these trajectories, as any memory effect present must only have a small effect on the nucleation rate. These results also suggest that the cluster size n is a good reaction coordinate candidate and so will be a good stating point for and comparison point to our machine learning predictions for the committor in chapter 6.

Chapter 4

Thermalisation in Seeding Methods

In this chapter we will implement and validate the two variations of seeding method introduced in section 2.6.3. Typically no distinction is drawn between these methods as work using them refers to the chosen method as "the seeding method" however as stated in section 2.6.3, and as we shall discuss in this chapter, there are differences between them. We will use our implementations to examine the assumption that the dynamics of the nucleus size metric n are "slow" relative to the other degrees of freedom, i.e. all other degrees of freedom are thermalised at all states of the nucleation process. We will then examine the role of this assumption in ensuring a correct implementation of these seeding methods.

4.1 Method Implementations

To examine seeding methods we need to first pick a system to use as the basis for our study. We choose crystal nucleation from the melt in a system of Lennard-Jones particles as this is a simple interaction potential and allows us to use reduced units for simplicity, as describe in section 2.3.2. While this choice removes the possibility of comparing our results to experiments, discrepancies in such comparisons are usually dominated by discussions of the interaction potential.

Before beginning to study thermalisation in seeding methods we first need to implement the two we intend to study. In order to verify that we have functioning implementations we will use the same system setup and method as reference [2] and

compare our calculated nucleation rates to those results for the single temperature and single seed methods described in references [2] and [33] respectively. This system uses the modified Lennard-Jones potential described in section 2.3.2 and the full details for our implementation using the LAMMPS MD engine [39] is given in appendix A.

4.1.1 Seed Generation

The first step in any seeding method is preparing the crystalline seeds. For this we need to generate systems with one solid cluster surrounded by liquid. This seems simple, however we must ensure that the solid phase, the liquid phase, and the interface between the two are all equilibrated at the same temperature. Ideally this temperature would be the temperature at which we want to run the system, however obtaining seed of exactly the same size equilibrated at different temperatures is practically very hard to achieve due to crystallisation of the liquid around the seed while cooling to the seed temperature. Because of this we equilibrate the seeds at one temperature as in [2] which we take to be $T_{\mathrm{EQ}} = 0.5$, this temperature is at a super cooling of $\Delta T = 0.017$ below the melting temperature $T_{\mathrm{m}} = 0.617$ [2].

To generate the seeds for our calculations we use the following methodology:

1. Generate an fcc box of LJ particles at the liquid density ρ_l.
2. Run the system at $T_{\mathrm{Melt}} = 0.8$ (NVT) to melt the solid block.
3. Run the system at T_{EQ} (NVT) to equilibrate to our desired temperature.
4. Define a sphere roughly the size of the seed required.
   (This gives 2 regions in or simulation, the seed and an outer region)
5. Remove all the particles the seed region
6. Insert LJ particles into the seed region at the solid density ρ_s
7. Minimise the energy of the inserted particles to remove potential overlaps with the liquid particles.
8. Run the seed particles at T_{EQ} (NVT).
9. Run the whole system at T_{EQ} (NPT).

While one region is being simulated the other region remains fixed, except of course for when simulating the whole system. We want to run the whole system for as little time as possible, but long enough for it to thermalise, as during this time our cluster size will be change from the originally desired size. The simulation times for the other regions are less important, however as $T_{\mathrm{EQ}} < T_m$, care must be taken to make sure the outer region is not run for long enough that nucleation events are likely. As discussed in section 2.4.1, in the rare event regime exceeding this time frame is usually not an issue. If $T_{\mathrm{EQ}} \sim T_m$ then this concern no longer matters however we must instead take care to simulate the inner region for as short a time as possible as the seed is very likely to shrink at this temperature.

We can plot the velocity distribution of the particles in each region, and the whole system, to check if they have thermalised properly. Figures 4.1, 4.2, 4.3 and 4.4 show the velocity distributions compared to the expected distribution at $T = 0.5$. The distributions containing all of the particle and only the liquid outer region both align very well with the theoretical distribution. The velocity distributions of the inner seed particles show a less well defined distribution as there are substantially fewer of these particles. We can calculate the kinetic temperature using the velocity distribution from the Maxwell-Boltzmann distribution. From this equation we can calculate T_{Kinetic} using the mean of the velocity distribution $\bar{v}$ as

$$T_{\mathrm{Kinetic}} = \frac{\bar{v}^2 m}{2\pi k_B}. \tag{4.1}$$

By calculating this kinetic temperature for each of our 3 regions we can check our systems are suitably equilibrated. Doing so we get $T_{\mathrm{K,all}} = 0.500 \pm 0.002$, $T_{\mathrm{K,Outer}} = 0.500 \pm 0.002$ and $T_{\mathrm{K,Inner}} = 0.48 \pm 0.06$. All of these are within error of T_{EQ} with the inner region having the largest error, which is likely due to there being few particles in this region.

4.1.2 Single Seed Method

As mentioned in section 2.6.3, for the single seed method, we take one seed and run it forward at multiple temperatures. For each temperature we run 106 trajectories using different random number sequences in order to give us an ensemble average with error bars. These average trajectories are shown in figure 4.5.

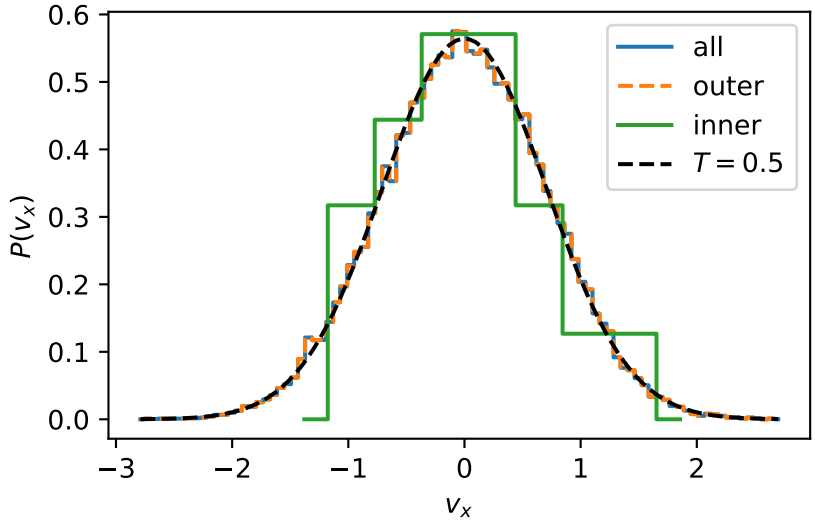


Figure 4.1: Distribution of velocity component, v_x after seed thermalisation for a 32125 particle system ($n_{\text{inner}} = 1128$ and $n_{\text{outer}} = 30872$) with an initial seed radius of $r = 4.2$ at a super cooling of $\Delta T = 0.017$. Dashed line shows the theoretical distribution for $T = 0.5$.

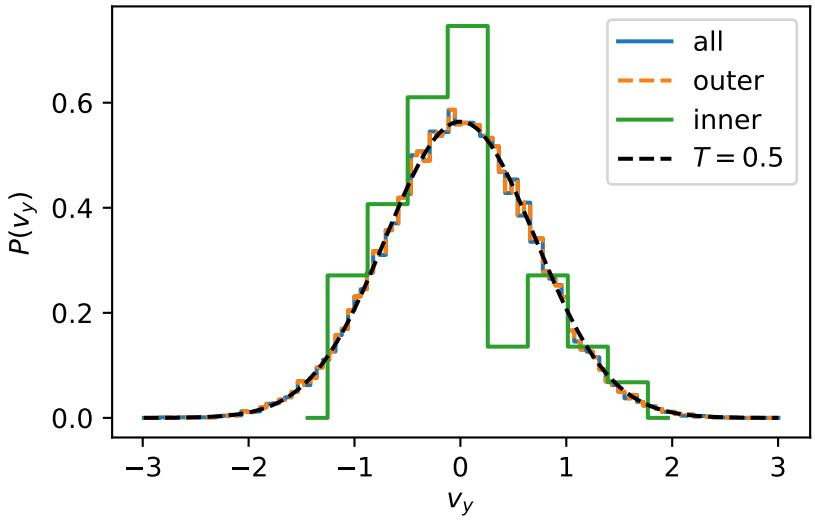


Figure 4.2: Distribution of velocity component, v_y after seed thermalisation for a 32125 particle system ($n_{\text{inner}} = 1128$ and $n_{\text{outer}} = 30872$) with an initial seed radius of $r = 4.2$ at a super cooling of $\Delta T = 0.017$. Dashed line shows the theoretical distribution for $T = 0.5$.

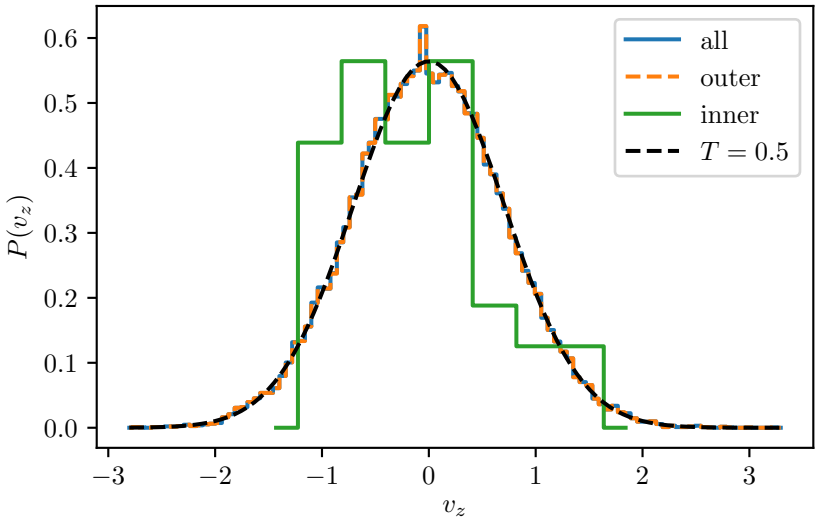


Figure 4.3: Distribution of velocity component, v_z after seed thermalisation for a 32125 particle system ($n_{\text{inner}} = 1128$ and $n_{\text{outer}} = 30872$) with an initial seed radius of $r = 4.2$ at a super cooling of $\Delta T = 0.017$. Dashed line shows the theoretical distribution for $T = 0.5$.

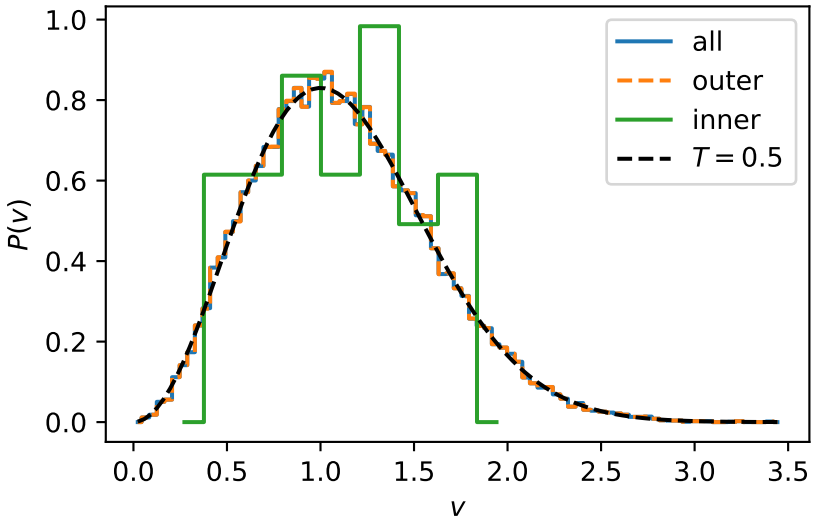


Figure 4.4: Distribution of particle velocity magnitude, v after seed thermalisation for a 32125 particle system ($n_{\text{inner}} = 1128$ and $n_{\text{outer}} = 30872$) with an initial seed radius of $r = 4.2$ at a super cooling of $\Delta T = 0.017$. Dashed line shows the theoretical distribution for $T = 0.5$.

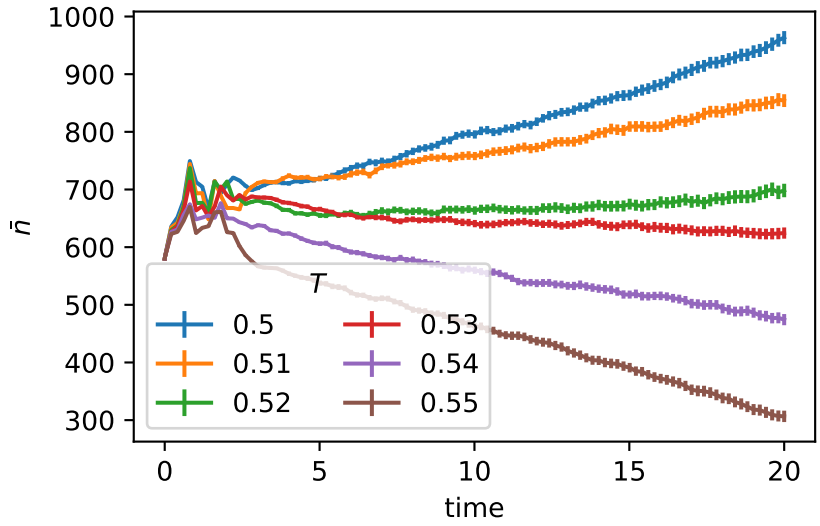


Figure 4.5: Ensemble average trajectories from a single seed at multiple temperatures. All trajectories are averaged over 106 runs with different random number seeds. Error bars are the standard error of the ensemble.

If we do a linear fit on each of the ensemble average trajectories in figure 4.5 then we can get an estimate of the gradient of the trajectories $\frac{d\bar{n}}{dt}\big|_T$. We are looking for the critical temperature T_c at which our chosen seed marks the top of the free energy barrier. At this temperature the seed will neither grow nor shrink on average so,

$$\left.\frac{d\bar{n}}{dt}\right|_{T_c} = 0. \tag{4.2}$$

Fitting the gradients of the trajectories against temperature allows us to calculate T_c for the starting seed. This method of finding T_c allows us to quantify the uncertainty in T_c from the trajectory ensembles, this fitting is show in figure 4.6.

At the start of the simulations there is some noise which sometimes means the linear fit of the trajectory no longer passes through the starting seed size n_{seed}. To account for the uncertainty generated by this drift we extend our linear fits back to time $t = 0$ giving the effective starting seed size $n_0'(T)$, instead of the simulation starting point n_0. We can then take a weighted average of $n_0'(T)$, giving more weight to trajectories with lower gradients to get an estimate of the effective starting seed

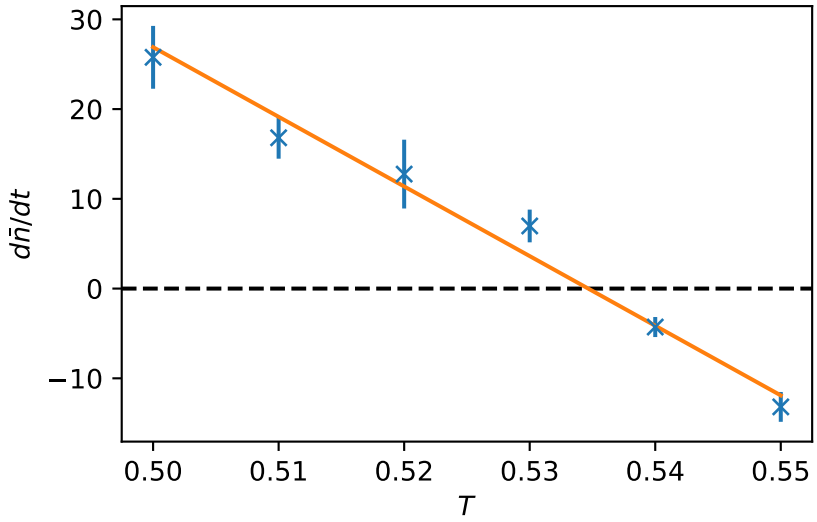


Figure 4.6: Gradients of the ensemble average trajectories used to find critical temperature for given seed size.

size,

$$n_c = \frac{\sum_0^{N_{\text{traj}}} n_0'(T_i)\, w_i}{\sum_0^{N_{\text{traj}}} w_i}, \tag{4.3}$$

$$w_i = \left(\frac{dn}{dt}\bigg|_{T_i} \right)^{-2}, \tag{4.4}$$

where N_{traj} is the number of trajectories.

We now have an estimate of the critical nucleus size at T_c. In order to calculate the CNT barrier height, using equation 2.21, we need to calculate the chemical potential $\Delta\mu$. While we could run these calculations ourselves, as the seeding method is our main point of interest, we simply use the values calculated by Espinosa et al. in [2]. To do so we simply fit a straight line to their results in order to interpolate between them, figure 4.7.

Now we have both of the elements in the nucleation rate exponent, equation 2.24. To get the Zeldovich factor, we can differentiate equation 2.19 and substitute in to 2.23, using equation 2.20 to evaluate at $n = n_c$. This give an alternative form

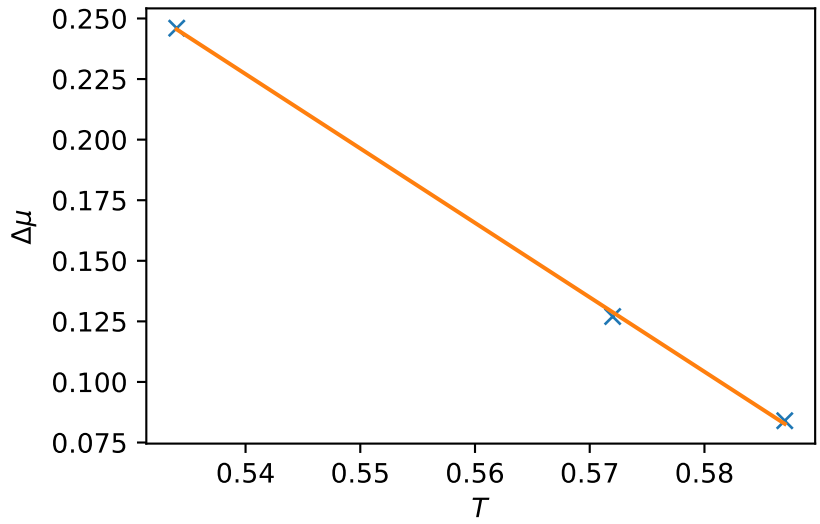


Figure 4.7: linear fit of the chemical potentials reported in [2], used to interpolate values of $|\Delta\mu|$ in this work

of the Zeldovich factor,

$$Z = \sqrt{\frac{\Delta\mu}{6\pi k_B T_c n_c}}. \tag{4.5}$$

Getting the liquid density, ρ_f, is fairly simple, we just run a bulk liquid simulation at T_c. As mentioned in section 4.1.1, it is important to ensure the fluid is properly equilibrated first at $T > T_m$ to ensure the starting block is fully melted, then at the desired temperature before calculating ρ_f. The only remaining component required to calculate the nucleation rate is the diffusion coefficient. We do this by running a new trajectory at T_c. We can calculate the effective diffusion coefficient at the top of the free energy barrier using

$$D = \frac{1}{2}\frac{\langle \delta n^2 \rangle}{t}, \tag{4.6}$$

where δn^2 is the squared mean change in cluster size $\delta n^2 = [n(t) - \bar{n}]^2$. By fitting δn^2 for the trajectory at T_c we can extract the diffusion coefficient, figure 4.8.

This gives us all of the components to calculate the nucleation rate using the single seed method. The comparison of nucleation rate from [2] and our results are shown in figure 4.9, and some of the intermediated components in table 4.1. The

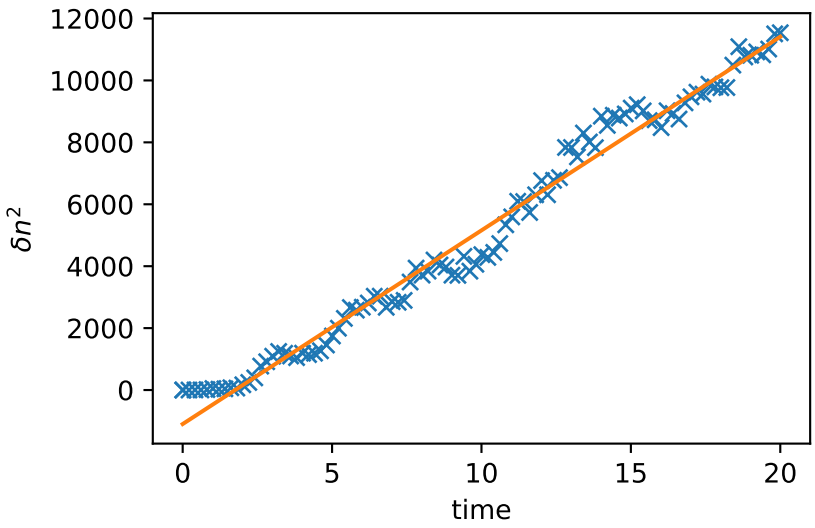


Figure 4.8: Squared displacement from mean for an ensemble of 106 trajectories with a seed of size $n_0 = 631$ at $T = 0.52$.

errors for the nucleation rate and other intermediate parameters are generated by propagating the errors from the initial fits through the calculations. The uncertainty in the temperature is similar for all of the data points, however the uncertainty in the nucleation rate grows rapidly at higher temperatures. This change in uncertainty is due to the rapid rate with which the nucleation rate varies at higher temperatures. This rapid uncertainty growth means that the precision with which we calculate T_c has a larger effect on the precision with which we can calculate the nucleation rate using this method.

4.1.3 Single Temperature Method

Another variation of seeding method is the single temperature method, in this section we will discuss its implementation as described in reference [33]. In this method we take several seeds and run them at one temperature. As with the single seed method, the aim is to find the conditions, seed size in this case, at which the ensemble average trajectory is flat, that is the seed neither grows nor shrinks on average. As in the single seed method we launch multiple trajectories form each starting point and calculate an ensemble average trajectory with errors, figure 4.10. For each starting

N_c	T_c	ρ	$\lvert\Delta\mu\rvert$	D	F_c	$\log_{10}(J)$
585	0.534	0.868	0.246	40	72	-32
3794	0.572	0.851	0.127	512	242	-106
12672	0.587	0.843	0.084	650	533	-234

(a) From reference [2]

N_c	T_c	ρ	$\lvert\Delta\mu\rvert$	D	F_c	$\log_{10}(J)$
631(1)	0.534(1)	0.869(1)	0.245(3)	313(1)	77(5)	$-29(3)$
3414(1)	0.567(1)	0.852(1)	0.144(4)	653(2)	250(20)	$-100(10)$
11272(1)	0.588(1)	0.842(1)	0.080(4)	1304(5)	500(100)	$-190(70)$

(b) From this work. Statistical errors show in parentheses

Table 4.1: Tables of results from reference [2] and this work.

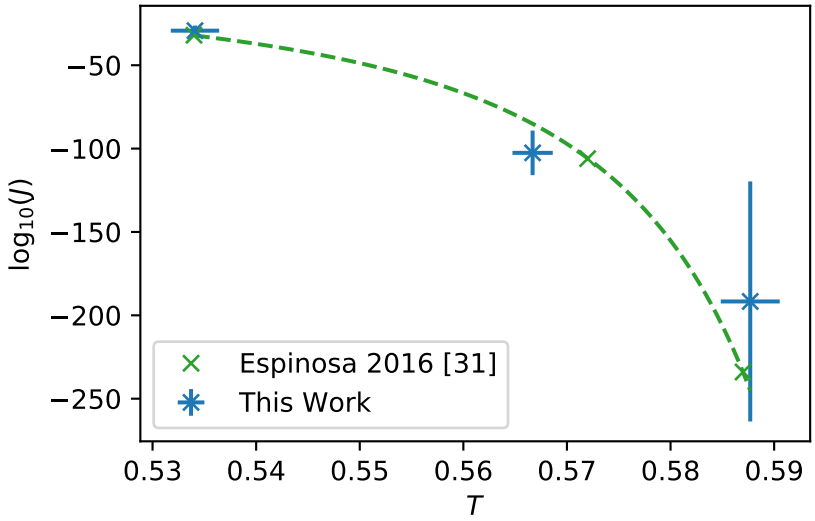


Figure 4.9: Nucleation rates using this seeding method implementation compared to the results from [2] which it is based on. The values in reference [2] are not presented with errors so no error bars are included.

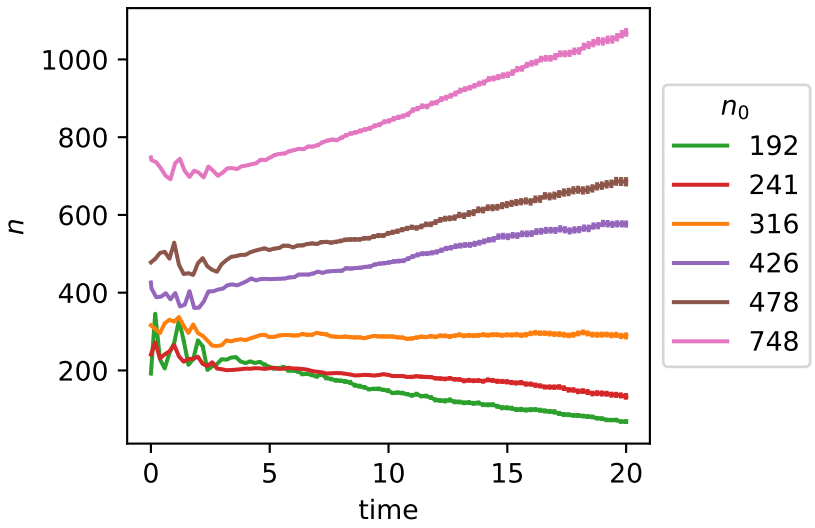


Figure 4.10: Ensemble average trajectories for single temperature method at $T = 0.53$. Each ensemble average is taken over 106 individual trajectories run from the same starting conditions with a different random number sequence.

seed n_{0i} we fit a straight line to the trajectory and extrapolate back to $t = 0$ giving a new effective starting point n'_{0i}. As before, our reasoning for this is to remove any initial noise in the trajectory. We then fit these effective starting points against the trajectory gradients to estimate the starting seed size at which the trajectory would be flat, n_c, shown in figure 4.11.

Ideally we would run another ensemble of trajectories starting from the critical seed size and calculate D in the same way as for the single seed method. However, as discussed in section 4.1.1, generating seeds is not trivial and it can be very time consuming to get a seed of the ideal size. The relation we are using to calculate D, equation 4.6, is only valid for pure diffusion, that is diffusion with no drift. This means we can only use it when the free energy has a zero gradient, which is true at the top of the barrier. To avoid having to generate new seeds we calculate values of D_i for each of the trajectories and take a weighed average based on the average growth rate $\frac{d\bar{n}}{dt}$ to mitigate against the pure diffusion assumption being invalid away

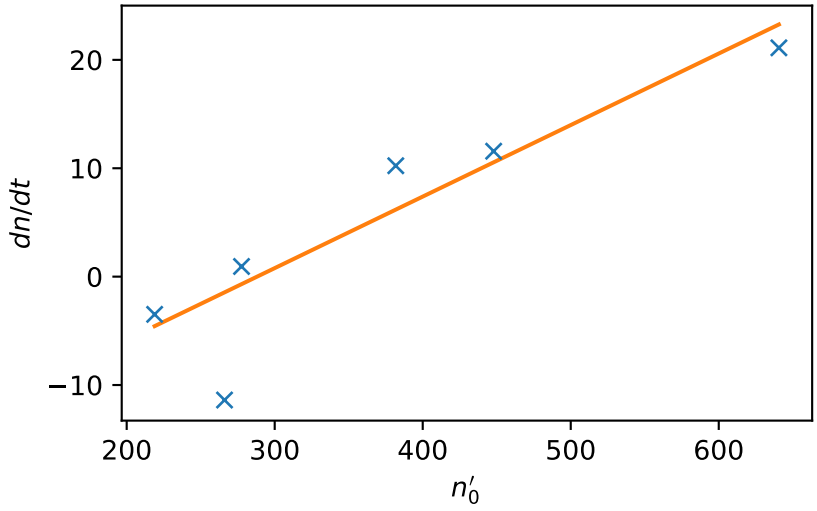


Figure 4.11: Fit of ensemble average gradients to get n_c at $T = 0.53$. The standard errors for these point are too small to be visible on the plots scale.

from the top of the barrier. This gives us an approximation for D of,

$$\bar{D} = \frac{\sum_{i=0}^{N_{\text{traj}}} D_i w_i}{\sum_{i=0}^{N_{\text{traj}}} w_i}, \tag{4.7}$$

$$w_i = \left(\left\langle \frac{dn}{dt} \right\rangle_i \right)^{-2}. \tag{4.8}$$

Where N_{traj} is the number of ensemble average trajectories.

Now we have an estimate of D we can use it to compute the gradient of the free energy using the trajectory growth rates [33],

$$\frac{\partial F(n)}{\partial n} = -\frac{k_B T}{D} \frac{d\bar{n}}{dt}. \tag{4.9}$$

We can use this, along with the partial differential of equation 2.19

$$\frac{\partial F(n)}{\partial n} = \left(\frac{2}{3}\right) n^{-\frac{1}{3}} \phi\gamma - |\Delta\mu|, \tag{4.10}$$

to fit for our two free parameters $\phi\gamma$ and $|\Delta\mu|$, as shown in figure 4.12. In this fit

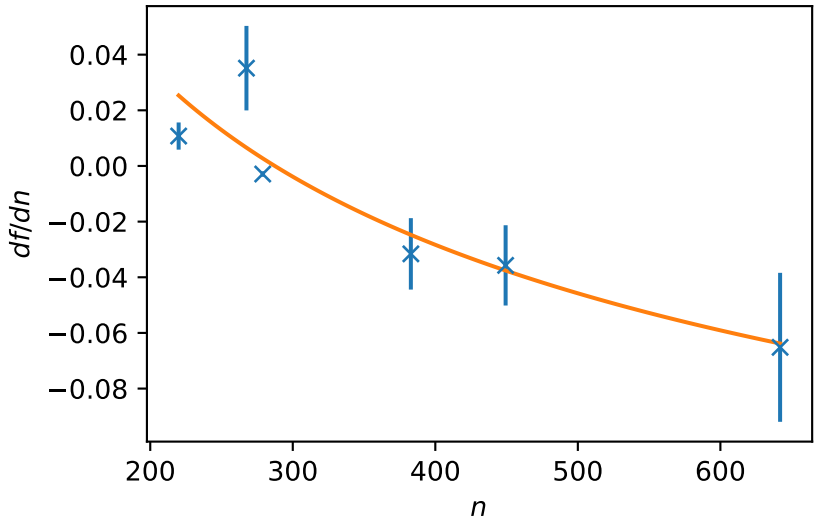


Figure 4.12: Fit of CNT free energy gradient $\frac{dF(n)}{dn} = \frac{2}{3}n^{-\frac{1}{3}}\phi\gamma - |\Delta\mu|$ using gradients calculated from growth rates $\frac{dF(n)}{dn} = -\frac{k_B T}{D}\frac{dn}{dt}$ to extract free energy parameters $\phi\gamma$ and $|\Delta\mu|$. This data is for a temperature of $T = 0.53$, and gives $\phi\gamma = 2.7 \pm 0.1$ and $|\Delta\mu| = 0.272 \pm 0.009$.

N_c	T_c	ρ	$\lvert\Delta\mu\rvert$	D	F_c	$\log_{10}(J)$
290(90)	0.53	0.869(1)	0.272(9)	170(70)	39(5)	−32(8)
400(100)	0.54	0.865(1)	0.28(1)	220(60)	52(6)	−41(9)

Table 4.2: Summary of results from the single temperature method.

we take the product $\phi\gamma$ as a single parameter instead of two separate ones as they only appear as this product.

We can use the parameters recovered from figure 4.12 to reconstruct the free energy barrier, figure 4.13. Again we now have all of the components to calculate the nucleation rate using equation 2.24. We can repeat this process with trajectories generated at different temperatures, re-using our seeds where possible, to generate the nucleation rate plot as we did for the single seed method. Figure 4.14 shows these results alongside the single seed methods results and the result from [2].

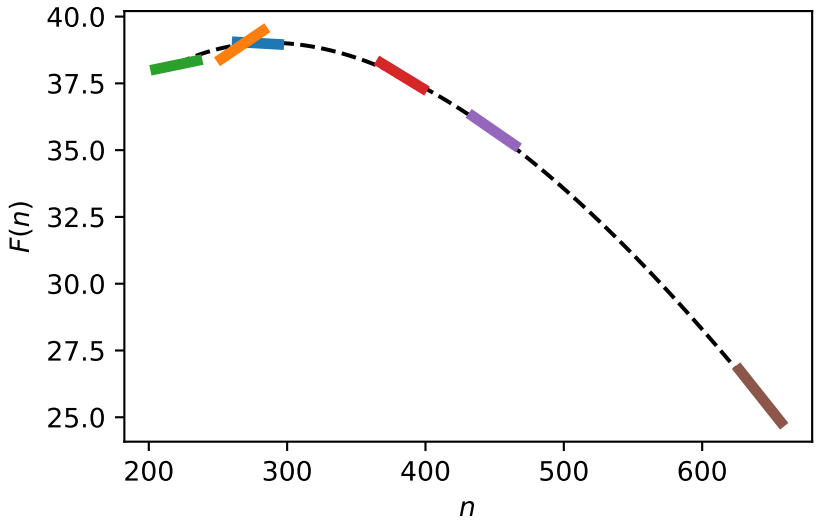


Figure 4.13: Free energy barrier at $T = 0.53$ generated from fitting gradients of the free energy measured via seeding. Coloured lines show the gradient of the free energy, $\frac{dF(n)}{dn}$, at the tangent point.

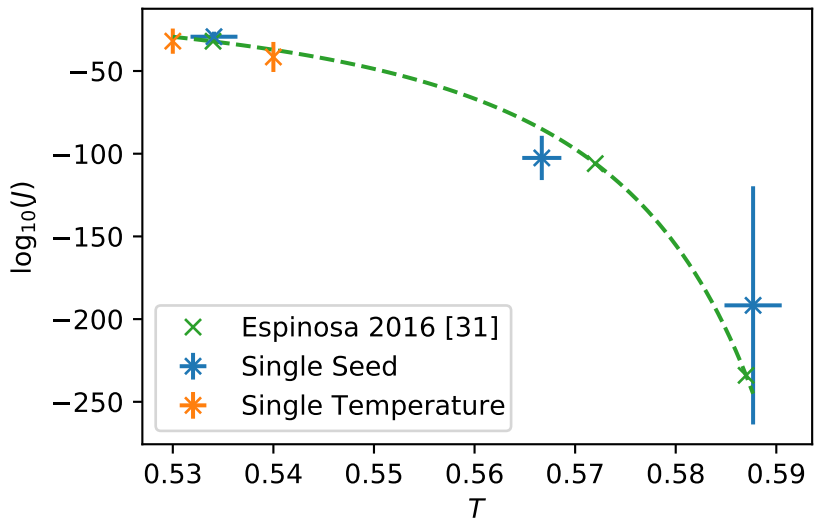


Figure 4.14: Collection of nucleation rates from the two seeding method variants.

4.1.4 Method Comparison

Both of these methods give similar results, but slightly differing processes with slightly different assumptions. The single seed method requires only one seed which is then run at multiple temperatures to find the temperature at which it is critical T_c, this same seed is then run at T_c to calculate the diffusion coefficient D. This method is efficient in terms of seed usage as we only need to generate one crystal seed to calculate a single nucleation rate, however we must choose the temperature for the nucleation rate calculation indirectly through the seed size. As this method requires running the same seed at multiple temperatures the nucleus size trajectories can be unstable as the thermostat temperature changes at the start of the trajectory. This means that fitting the linear sections of the growth rates does not necessarily, and in our case did not, extrapolate back to this starting cluster size n_0, for this reason we have some uncertainty on the critical nucleus size n_c. This method also requires us to calculate the chemical potential $|\Delta\mu|$ and the liquid density ρ_l via some other process.

The single temperature method uses a selection of seeds to estimate the critical nucleus size n_c at a given temperature T. As each seed is only run at a single temperature the seed and surrounding liquid could be thermalised at this temperature to reduce the instability at the start of the trajectory, however as we wanted to re-use the seeds at multiple temperatures we did not do this. If we did generate the seeds at the chosen temperature T then we would have already needed to calculate ρ_l at this temperature and as we estimate all other parameters for the free energy and nucleation rate from the nucleus size trajectories we do not require any extra simulations to calculate the nucleation rate J. The method we employ for calculating D does add a large degree of uncertainty as we calculate it in regions of the free energy landscape where there may not be pure diffusion. This method also requires us to calculate the free energy gradient from growth rates, this means that the trajectories averaged to generate this must properly sample the landscape in both growing and shrinking directions. While this is likely near the top of the barrier, the further from n_c we go, the less likely it becomes.

For the single temperature method we only computed nucleation rates for the lower temperatures, relating to smaller critical nuclei and therefore smaller simulation boxes. When running the calculations for larger systems we saw a dependence on the size of the simulation box, i.e. we calculated noticeably different average

growth rates for similar seeds in boxes of different sizes. We believe this is a result of secondary nuclei in the larger systems. When generating the seeds, we have to melt the initially generated starting system. During this process we monitor the largest cluster size which fluctuates as the system equilibrates to our chosen temperature for melting. In larger systems this process takes longer and the fluctuations are larger. After the system is melted, we equilibrate to our equilibration temperature, which again takes longer for larger systems, however the longer we spend equilibrating the larger any residual nuclei grow. This means when we add our central seed there are likely to be other, smaller, nuclei in our simulation box. The effect of these nuclei could be leading to the effect on the growth rate in these systems, and the effect of multiple nuclei on these methods would be an interesting area of further study, which we will not cover in this work.

4.2 Heat Transport

In all of the simulations so far, the thermostat is controlling the velocities of all particles. This means we force the system to thermalise to the simulation temperature, the speed at which it thermalises is controlled by the damping parameters in the thermostat. This means any heat transport in the system is restricted to a short timescale, heat is removed or inserted via the thermostat to maintain a constant a uniform temperature on a timescale faster than it can diffuse though the simulation box. In a system with a growing or shrinking seed, heat will be released at the seed boundary, if the heat source for the system is in the far field then the thermal energy must be transported to or from that boundary. This effect is removed if the thermostat is applied to the whole system.

Our aim in this section is to establish the relative timescales of heat transport to or from a growing or shrinking nucleus and the growth of that nucleus, as well as the range over which the temperature profile around a single growing or shrinking nucleus is non-uniform. We also examine, without a global thermostat, the extent to which the $n(t)$ trajectories sampled at each nucleus size represent an ensemble in which the heat flow to or from the nucleus is zero and all other degrees of freedom are thermalised.

4.2.1 Heat Flux From a Seed

To demonstrate the non-uniform temperature profile, which results from the release of latent heat during the growth of a nucleus, we divide our simulation into shells centred on the starting seed and calculate the kinetic temperature for each shell. Figure 4.15 shows the radial temperature profile of the system, at several time steps. This system is thermalised at $T_{\mathrm{EQ}} = 0.534$ and the thermostat released at $t = 0$.

This figure shows the radial temperature distribution for time $t = 1.22$. This means there has been some time for the temperature to become non uniform as the nucleus changes size during this time. From this we see the temperature flowing from the seed surface, $r \approx 5$, outwards. The changes in temperature decrease at larger radii as there are more particles in each of the shells meaning the thermal energy is spread between more particles. It is worth noting that this figure shows time steps starting from $t = 1.22$, this means the nucleus has had time to change size generating a temperature profile. Also these time steps were selected solely because they offer a clear view of the heat transport as the seed is growing at a constant rate during the interval they span.

This heat transport may have an effect on the growth rate of a system, if the local temperature around a cluster is different from that of a thermostatted region positioned a distance away from the seed.

4.2.2 Thermal Diffusivity

To better understand the flow of heat to and from a seeded nucleus, it would be beneficial to calculate the timescale of heat transport through the surrounding liquid. This can be used to inform the placement of the thermostatted region such that we can simulate growth under non-equilibrium steady-state conditions. To do this, we want to calculate the thermal diffusivity of our Lennard-Jones system. Values for the thermal diffusivity are available, but not for the modified Lennard-Jones potential used in this work.

We begin by calculating the heat current vector for the bulk liquid. The heat current vector is defined mathematically as [40],

$$\mathbf{q} = \frac{1}{V}\frac{d}{dt}\sum_i E_i \mathbf{x}_i, \tag{4.11}$$

where V is the system volume, $\mathbf{x}_i$ is the position of particle i, and E_i is the energy

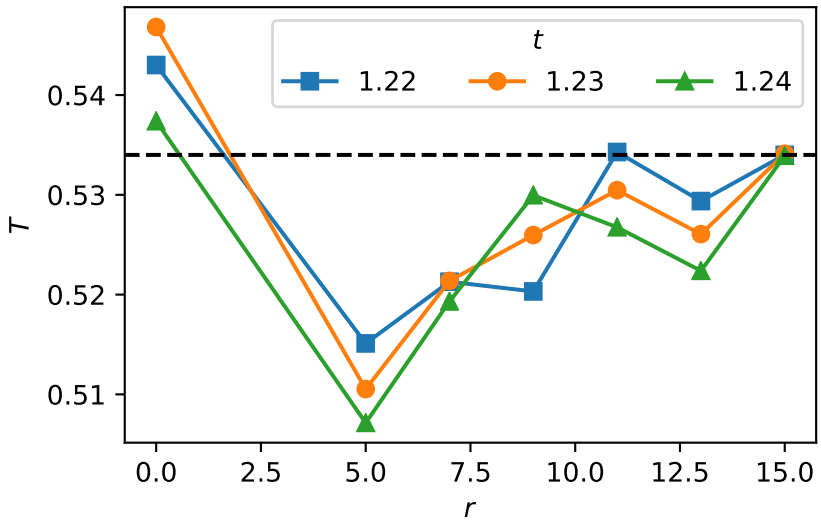


Figure 4.15: Radial temperature profile for a system thermalised at $T = 0.534$ (dashed line) with a starting seed size of $n = 497$ ($r_{\text{seed}} \approx 5$). System is run with an Nosé Hoover barostat and no thermostat (i.e NPH ensemble) from $t = 0$. Each data point is the temperature in a shell with extents $r < 4$, $4 \leq r < 6$, $6 \leq r8$, $8 \leq r < 10$, $10 \leq r < 12$, $12 \leq r < 14$ and $r \geq 14$.

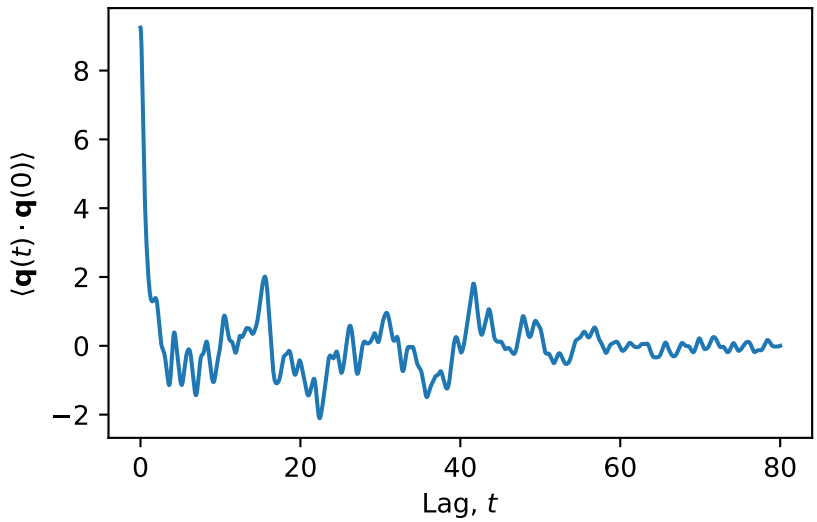


Figure 4.16: Autocorrelation function of the heat current vector $\mathbf{q}$ for a NPT bulk liquid Lennard-Jones system at $T = 0.534$, $P = -0.02$.

of particle i. While this could be computed post simulation, the heat current can be computed natively in LAMMPS using the `compute heat/flux` command. We can then use the Green-Kubo relation [40],

$$k = \frac{V}{3k_B T^2} \int_0^\infty \langle \mathbf{q}(t) \cdot \mathbf{q}(0) \rangle \, dt, \tag{4.12}$$

to calculate the thermal conductivity of the system from the heat currents obtained in the MD simulation. Here $\langle \mathbf{q}(t) \cdot \mathbf{q}(0) \rangle$ is the autocorrelation function of the heat current vector, shown in figure 4.16. Using 10 MD simulations with different random number sequences, we obtain a thermal conductivity of $k = 6.73 \pm 0.01$.

To calculate the thermal diffusion, we also need the specific heat capacity at constant pressure. This is calculated by simulating a bulk liquid system under a NPT thermostat at several temperatures and taking the gradient of the energy with respect to temperature, shown in figure 4.17. From this we get a heat capacity of $c_p = 5.27 \pm 0.01$. The last parameter required to calculate the thermal diffusivity is the liquid density which we calculate from our simulations as in section 4.1.2, to be $\rho_{\text{liquid}} = 0.869 \pm 0.001$. This finally allows us to calculate the thermal diffusivity

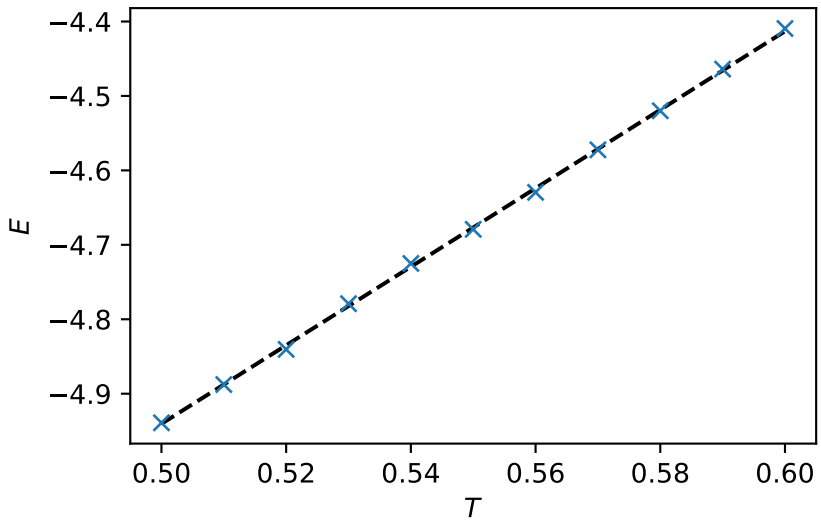


Figure 4.17: Energy temperature relation for bulk liquid Lennard-Jones simulation at constant pressure to give a specific heat capacity of $c_p = 5.27 \pm 0.01$.

using [40],

$$D_T = \frac{k}{\rho_{\text{liquid}} c_p}, \tag{4.13}$$

to be $D_T = 1.468 \pm 0.004$. These values of k, c_p and D_T are similar to those found in the literature [41, 42, 43] although each uses a different variant of the Lennard-Jones potential.

4.2.3 Latent Heat

If we calculate the latent heat capacity of our system, by taking the energy difference between the solid and liquid phases, we can compute the energy released by adding one particle to the cluster. We can use that energy change to setup a radial heat transport simulation to compute the time taken for the energy input at the centre to propagate outwards. Form our simulations we calculate the latent heat capacity to be $L = 0.89 \pm 0.01$. This gives a temperature change of $\delta T = 0.169 \pm 0.002$ which we use to setup our heat transport simulation. δT is the temperature change if a single particle is given the energy increase of L, $\delta T = L/c_p$. The heat transport simulation is set up with the temperature initially at $T = 0.534$ for all radii except

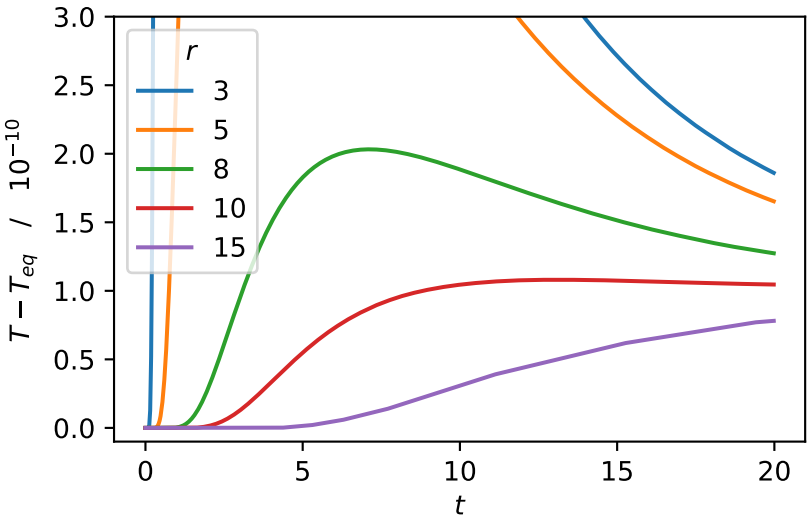


Figure 4.18: Temperature changes for simulated heat transport from numerical integration of the radial heat equation. The initial condition for is a uniform temperature of $T_{eq} = 0.534$ with an additional $\delta T = 0.169$ at $r = 0$.

for at the centre, $r = 0$ where we increment it by δT. We can then evolve this system by numerically integrating the radial heat equation,

$$\frac{\partial T}{\partial t} = D_T \frac{1}{r^2} \frac{\partial}{\partial r} \left(r^2 \frac{\partial T}{\partial r} \right). \tag{4.14}$$

Figure 4.18 shows the temperature changing with time at different radii and, as we would expect from this system, the temperature at all radii are converging towards a temperature sightly larger than T_{eq}. If we look at the $r = 8$ curve we can see that it peaks at $t \approx 7$, but the temperature at this radius is effected at a much earlier time. Given this we can say that a thermostat at $r = 8$ will take about $1 - 2$ time units to "react" to a temperature change at $r = 0$. This simple simulation differs from what we expect from a seeded MD simulation as here we only have one particle attaching to the nucleus whereas in a seeded MD simulation there would be a continuous flux of particles attaching and detaching as well as potentially other interaction in the bulk.

We can also consider the temperature change of the entire Lennard-Jones system expected given the calculated change in cluster size. For our system with

no thermostat from section 4.2.1, the net change in cluster size is $\delta n = -30$, so the cluster shrank over the cause of the simulation. Given our latent heat calculations this should introduce an energy change of $\delta E = -(26.8 \pm 0.3)$, which should lead to a temperature change of $\delta T_{\text{Theory}} = (-4.42 \pm 0.05) \times 10^{-3}$. The net temperature change in the MD simulations from section 4.2.1 is $\delta T_{\text{MD}} = (-2.08 \pm 0.02) \times 10^{-3}$. While this is half the predicted value, given the naivety of the calculation, the result being the same order of magnitude is promising.

4.2.4 Thermalisation

When we use CNT, we assume that the system is in quasi-equilibrium, that is all other degrees of freedom thermalise on the timescale on which the cluster size changes. Given that we expect heat to move in our system it seems likely that this non-equilibrium heat flow could cause problems. To check if this assumption is still valid in systems where latent heat is accounted for, we will examine whether or not the size of largest cluster is a slow degree of freedom compared to the particle velocities. We want to check that at each cluster size, the velocity distribution of the particles in the system are what we would expect for the equilibrium distribution.

To generate these distributions we run simulations on the same systems with a few small distinctions. For this system we equilibrate to a temperature of $T_{\text{EQ}} = 0.534$ and equilibrate our seed using the same seed radius as the critical cluster size used in section 4.1.2, however due to variation in the equilibration process, this produced a starting seed of $n_0 = 497$. This seed is sub critical ($n_c = 631 \pm 1$), but still allows us to test the distributions. When running these simulations we record the positions and velocity components of all particles in the system ($N = 32150$) along with the size of largest cluster at every simulation step $\tau = 0.0002$. We repeat these simulations 10 times with different random number seeds to generate different trajectories. We also repeat this for 3 thermostat conditions thermostating all particles, thermostating particles at a radius from the centre of $r > 8$, and thermostating none of the particles. These 3 condition represent a global thermostat, a thermostatted region nearby the seed, and a thermostatted region in the far field (i.e no thermostat).

The particle positions were extracted using the LAMMPS `dump custom` command, and the global state parameters such as largest cluster size were outputted to the usual log file. Given the amount of data generated and the links between

entries in log and dump files some processing was required before we could use the data. To get this data into a useable form, we implemented a parser in C to extract this data from the multiple log and dump files into an SQLite database for easier manipulation (for an indication of scale the size of the complete database is around 800GB).

After generating this dataset, we build velocity distributions for each cluster size under each of the thermostat conditions. The distribution of the velocity components v_x, v_y, v_z and the speed $|v|$ for the starting cluster size of $n = 497$ are shown in figures 4.19, 4.20, 4.21 and 4.22. These distributions for all of the thermostat conditions align with the theoretical distributions given by Boltzmann distribution for the velocity components and the Maxwell-Boltzmann distribution for the speed. For this cluster size all of the distributions align with the expected theoretical distributions. This is unsurprising for the starting seed size, but holds true for all cluster sizes in the trajectories.

We can use the mean of these distributions and the functional form of the theoretical distributions to calculate a kinetic temperature for each cluster size n, shown in figure 4.23. These temperatures are closer to the defined simulation temperature for cluster sizes with more samples, figure 4.24. The kinetic temperature deviates most, and has the larger errors where the samples are all from trajectories which are only growing or shrinking, the fraction of samples from growing and shrinking trajectories is shown in figure 4.25. This deviation means that our assumption of thermalisation is least valid in these purely shrinking or purely growing cluster sizes. This is most important for the single temperature method, section 4.1.3, as it relies on the growth rate to estimate the free energy gradient. If the system is not thermalised at a given cluster size, then that estimation becomes invalid. This deviation could be reduced, and the assumption's validity restored, by generating more trajectories to get better sampling, however there will always be a practical limit as a cluster can only grow to a finite size in a finite time span. Also increasing the number of trajectories increases the computational cost which reduces one of the main benefits of seeding methods.

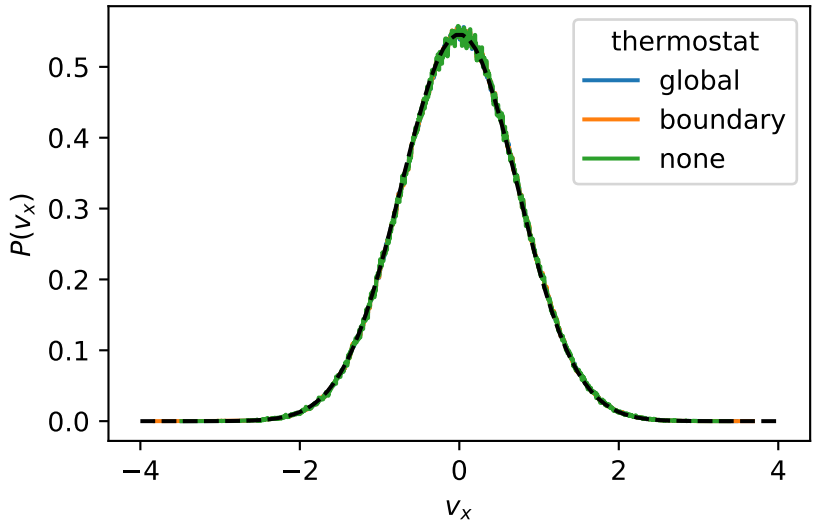


Figure 4.19: Distribution of x velocity components at $n = 497$ compared to the theoretical distribution (dashed line) for $T = 0.534$.

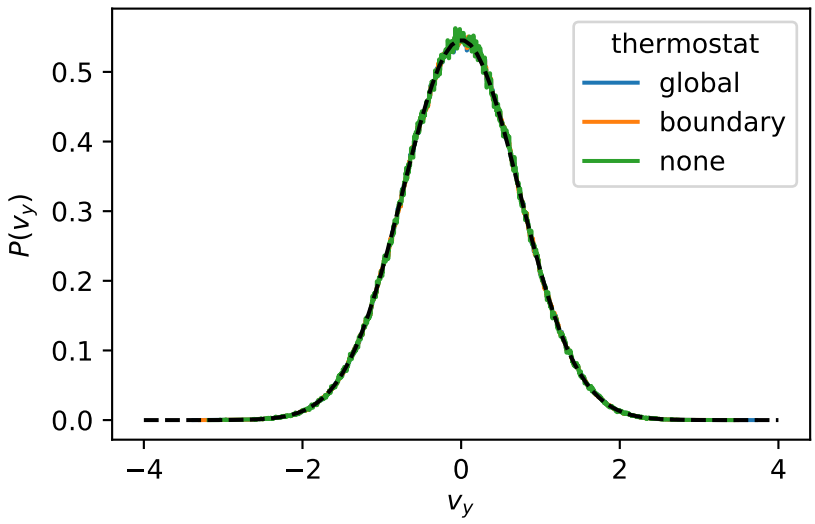


Figure 4.20: Distribution of y velocity components at $n = 497$ compared to the theoretical distribution (dashed line) for $T = 0.534$.

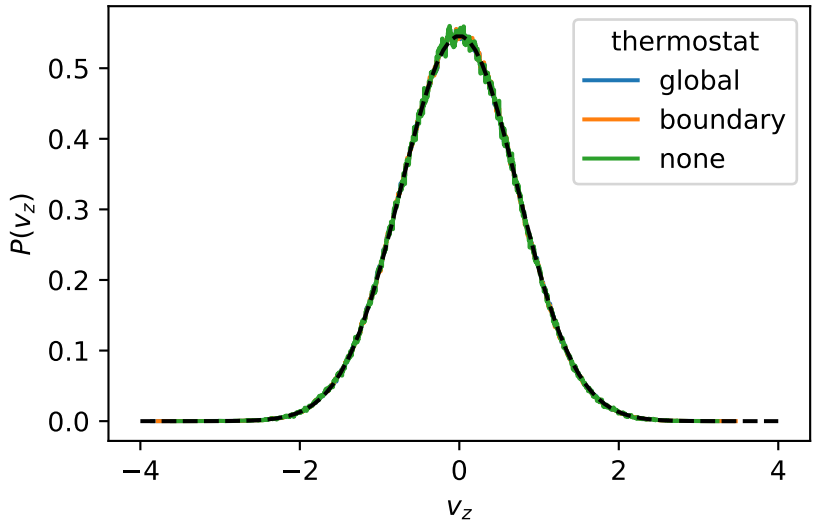


Figure 4.21: Distribution of z component of particle velocity at $n = 497$ compared to the theoretical distribution (dashed line) for $T = 0.534$.

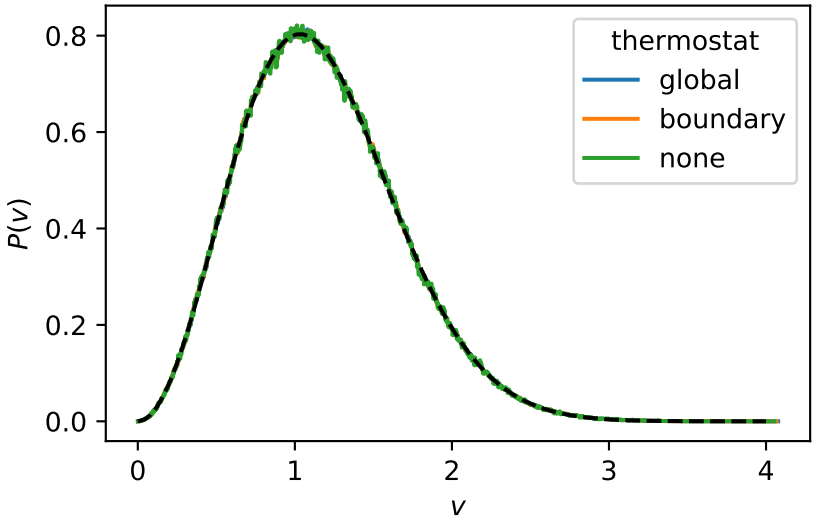


Figure 4.22: Distribution of particle speeds for cluster size $n = 497$ compared to theoretical distribution (dashed line) for $T = 0.534$.

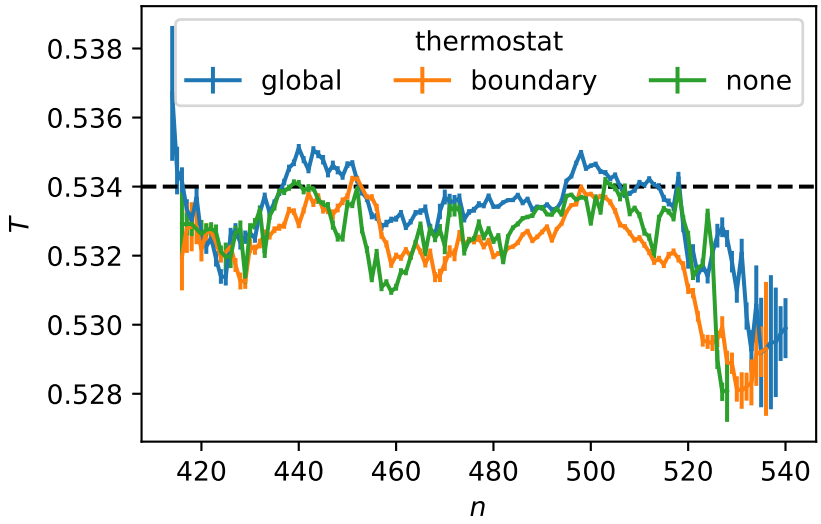


Figure 4.23: Kinetic temperature generated from all states with a given cluster size n for 3 thermostat conditions. With the boundary thermostat set for all particles $r \geq 8$ from the centre, and the dashed line indicating the simulation temperature $T_{\text{run}} = 0.534$.

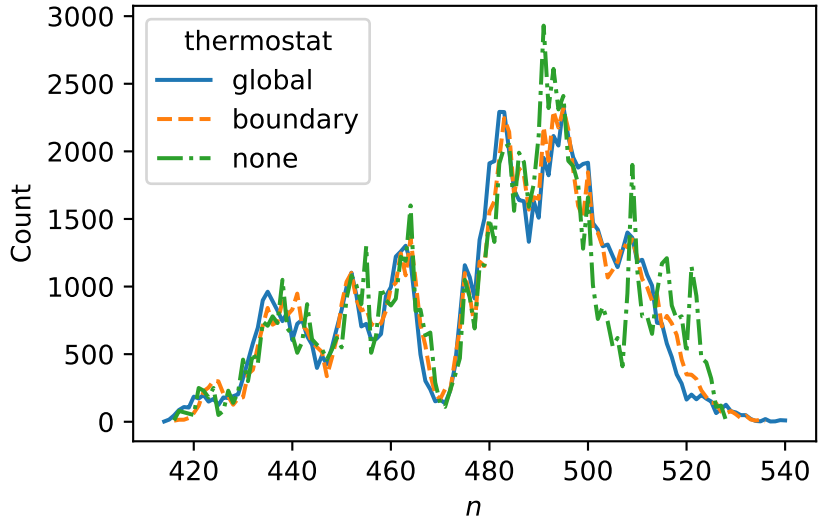


Figure 4.24: Number of samples for each recorded cluster size n.

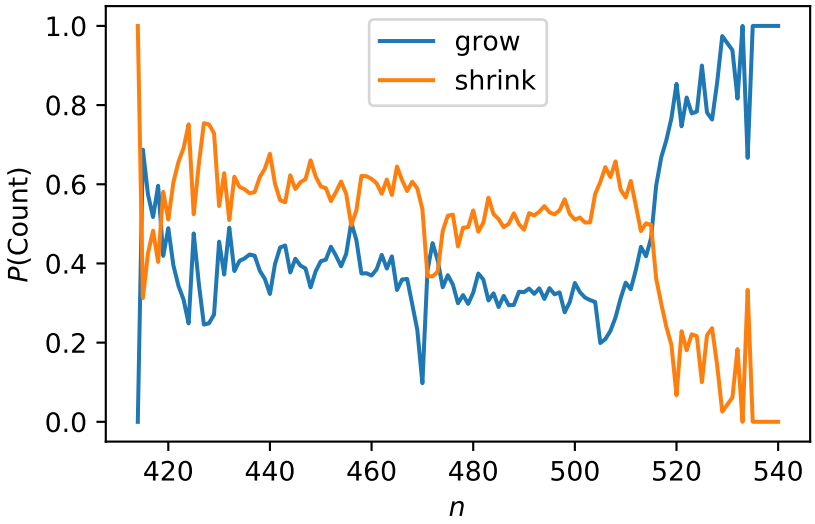


Figure 4.25: Fraction of samples at each cluster size for which the trajectory grows or shrinks on average for the globally thermostatted trajectories.

4.3 Conclusion

When comparing the two variants of the seeding method, our results for the single seed method align reasonably well with those reported by Espinosa et al. in [2]. At higher temperatures, the errors in the nucleation rate J increase substantially. This is to be expected as, in this regime, a small change in temperature will result in a large change in nucleation rate. The results for the single temperature method also fit with both the results from [2] and the single seed method, however it gave less precise values for the critical nucleus size and diffusion coefficient.

CNT relies on the system being in quasi-equilibrium as the cluster size changes or, in other words, that the cluster size is a slow degree of freedom relative to the thermalisation time of the system. Examining the velocity distributions for each cluster size shows this to be the case for our Lennard-Jones systems. Calculating the kinetic temperatures for these distributions showed that the deviation from the thermostat temperature was smallest at the most sampled sizes and worst in case where we did not sample from both growing and shrinking trajectories. Sampling from both growing and shrinking trajectories is most important for the single temperature seeding method as it relies on linking the growth rate of the cluster to

the gradient of the free energy barrier, which to correctly sample require trajectories travailing in both directions. The likelihood of a trajectory either growing or shrinking can be calculated from the committer, however calculating that is a more computationally expensive task.

Allowing for heat transport in these systems seems to have little effect on the velocity distributions of the systems on the short timescales required for seeding methods. However, local temperature changes and heat transport may have effects in longer timescale computations like path sampling methods. This is enforced for simulations which use a global thermostat, which is potentially unphysical, if the cluster growth is actually rapid in an open system. However the assumption within CNT is that the quasi-equilibrium distribution is recovered by averaging over an ensemble of configurations at each n which contain both shrinking and growing nuclei within the same population.

These results lead us to conclude that these seeding methods are consistent with the assumptions of CNT as long as we sample both growing and shrinking trajectories. To progress this work, full seeding nucleation rate calculations could be conducted using the three thermostating conditions discussed to to see if there is any significant difference between them. A more complex thermostating condition could also be applied to more realistically simulate the coupling of the system to a heat source/sink at different radii by using the radial heat equation to adjust the temperature near the boundary of the MD simulation to account for heat flow.

These results, and methods, could also be compared to results from path sampling calculations with similar temperatures and seed sizes. The path sampling results would provide another point of reference as they do not make the same assumptions as CNT, although they do make other assumptions and can have some reaction coordinate dependence. Path sampling calculations could also be used to further study the effects of heat transport by conducting them with different thermostat conditions. Because path sampling trajectories are longer than those used in our seeding calculations the heat transport effects should be larger.

A final suggestion for further study would be to apply these tests to other systems. As we discussed in section 4.1, we chose this Lennard-Jones system for its simplicity, but the effects may be different in other, more complex systems. However, any effect must be considered alongside the uncertainties that arise with increasing complexity.

Chapter 5

Memory Quantification and Identification

As described in section 2.4.2, CNT assumes that the dynamics of the cluster size n are Markovian. If incorrect, this assumption would invalidate predicted nucleation rates based on CNT, including those obtained via the seeding methods discussed in chapter 4. To validate if our seeding trajectories are Markovian, we will quantify the extent to which they can be reproduced by a 1D random walk on the free energy landscape given by the single temperature seeding method.

5.1 Comparison method

To determine if a trajectory $\mathbf{x}$ is best described by a Markovian or non-Markovian walk on a given free energy landscape, we calculate a figure of merit which quantifies the statistical confidence that a given sample trajectory could belong to a set of trajectories generated by a known stochastic reference process. For this figure of merit we calculate the likelihood of the trajectory by breaking the trajectory down into discrete moves $x_1 \to x_2$ and determining the likelihood of those moves given the free energy landscape and the type of dynamics $P(x_1 \to x_2) = P(x_2|x_1)$. Taking the product of this likelihood for all N_{steps} steps in $\mathbf{x}$ give us the likelihood for the trajectory,

$$L(\mathbf{x}) = \prod_{i=1}^{N_{\text{steps}}} P(x_{i+1}|x_i). \tag{5.1}$$

To avoid our figure of merit becoming unreasonably small for long trajectories, we will use the natural logarithm of the likelihood, which can be expressed as the sum of the individual step likelihoods,

$$\ln\left[L(\mathbf{x})\right] = \sum_{i=1}^{N_{\text{steps}}} \ln\left[P(x_{i+1}|x_i)\right]. \tag{5.2}$$

As these processes are stochastic, the steps within each walk is dependent on the random numbers rolled, hence these random numbers could heavily effect the log likelihood score. To mitigate against this we will consider an ensemble of N_{traj} trajectories $\mathbf{x}_i$, allowing us to compare ensembles based on the average,

$$\ell = \frac{1}{N_{\text{traj}}} \sum_{i=0}^{N_{\text{traj}}} \ln\left[L\left(\mathbf{x}_i\right)\right], \tag{5.3}$$

and standard error,

$$\epsilon_\ell = \frac{1}{N_{\text{traj}}} \sqrt{\sum_{i=0}^{N_{\text{traj}}} \left\{\ln\left[L\left(\mathbf{x}_i\right)\right] - \ell\right\}^2}. \tag{5.4}$$

5.1.1 Reference Stochastic Processes

Before using this comparison on our system of interest, we must first ensure it can distinguish between known stochastic processes which we define with and without memory. For the Markovian case we use a discrete time 1D random walk on a energy landscape, E, with equal probability of attempting to move in the positive or negative direction ($P_{\text{Attempt}}(+) = P_{\text{Attempt}}(-) = \frac{1}{2}$) and take the Boltzmann factor of the move as the acceptance probability ($P_{\text{Accept}} = \exp\left\{-\beta\Delta E\right\}$), similar to the description in section 2.1.1. We must be more careful when we make our choices for the non-Markovian case so that we ensure we still satisfy detailed balance.

To generate a discrete time random walk with memory we take an approach similar to adding momentum to the random walk, this can be thought of as preferring to continue moving in the same direction. For this we define a memory function $\mathcal{M}(m)$ which applies some weight to the current memory of the system m. Here m is a sum of the previous N_M moves, ± 1 for moves accepted attempts and 0 for rejected attempts, where N_M is the defined memory length. For example, the

memory at the i^{th} step of trajectory $\mathbf{x}$, m_i is given by,

$$m_i = \sum_{j=i}^{\max(0,i-N_M)} [x_j - x_{j-1}]\,. \tag{5.5}$$

We will discuss the form of this memory function later, equations 5.11 and 5.12. To allow this memory to affect the random walk, we adjust the probabilities of attempting to move in either direction. The probability of attempting to move in the positive direction, given some current memory m is given by,

$$P_{\text{Attempt}}(+|m) = \frac{1}{2}\left[1 + \mathcal{M}(m)\right]. \tag{5.6}$$

We must now adjust our acceptance probability so that we satisfy detailed balance, that is, the probability of choosing to move from state x_1 to an adjacent state x_2 and accepting that move must be equal to the time reversed probability of choosing to move from x_2 to x_1 and accepting that move,

$$P(x_1)P_{\text{Attempt}}(x_2|x_1)P_{\text{Accept}}(x_2|x_1) = P'(x_2)P'_{\text{Attempt}}(x_1|x_2)P'_{\text{Accept}}(x_1|x_2). \tag{5.7}$$

Under time reversal, the sign of the memory state m will be flipped as any move to the right forwards in time would be a move to the left backwards in time. A simple choice of acceptance probability, and the one we shall use, is

$$P_{\text{Accept}}(x+1|x,m) = \frac{1}{1+\mathcal{M}(m)} \exp\left\{-\beta\Delta E\right\}. \tag{5.8}$$

Here the acceptance probability is dependent on x because of the implicit x dependence in ΔE, unlike the attempt probability which only depends on the current memory m.

These definitions of selection and acceptance probabilities are for moving in the positive direction. Any moves in this direction will increase m. We can also define the probabilities for moving in the opposite direction,

$$P_{\text{Attempt}}(-|m) = \frac{1}{2}\left[1 - \mathcal{M}(m)\right], \tag{5.9}$$

$$P_{\text{Accept}}(x-1|x,m) = \frac{1}{1-\mathcal{M}(m)} \exp\left\{-\beta\Delta E\right\}. \tag{5.10}$$

With these definitions of choosing and accepting moves we can construct our toy model, described in algorithm 1.

Algorithm 1 Pseudo-code for a random walk with memory, m. Here m is the sum of the last N_M moves which is updated each step, and rand is a uniformly distributed random number in the range $[0, 1]$.

```
x = x_start
m = 0
for number of steps do
    if rand < P_Attempt(+|m) then
        move = +1
    else
        move = −1
    end if
    if rand < P_Accept(x + move|x, m) then
        x = x + move
    end if
    update m
end for
```

This simple model allows us to generate random walks on any given energy landscape in order to assess the performance of our likelihood analysis. For the purposes of testing we consider 3 different energy landscapes. The first of these landscapes is the trivial case of a flat energy landscape, here the walker is free to roam across the landscape with no external forces impacting it. The second choice is a simple quadratic potential, here the walker is strongly confined to a small region of the landscape. The third choice is a CNT style energy landscape of the same form as the free energy landscape defined in section 2.4.2 equation 2.19 and shown in figure 5.1, this should allow us to see the best case performance of our methodology when applied to CNT trajectories. With the CNT landscape we set a minimum value of $x = 0$ and reject any moves which would lead to values $x < 0$.

So far we have discussed selection and acceptance probabilities, our 1D random walk algorithm, and the energy landscapes on which we intend to conduct our tests, but there is one further thing we must discuss before running our tests, that is the memory function $\mathcal{M}$. As mentioned previously $\mathcal{M}$ controls the amount of effect that the memory has on the system. In our tests we will consider two forms of $\mathcal{M}$,

$$\mathcal{M}_L(m) = \xi m, \tag{5.11}$$

$$\mathcal{M}_R(m) = \frac{\xi m}{N_M}, \tag{5.12}$$

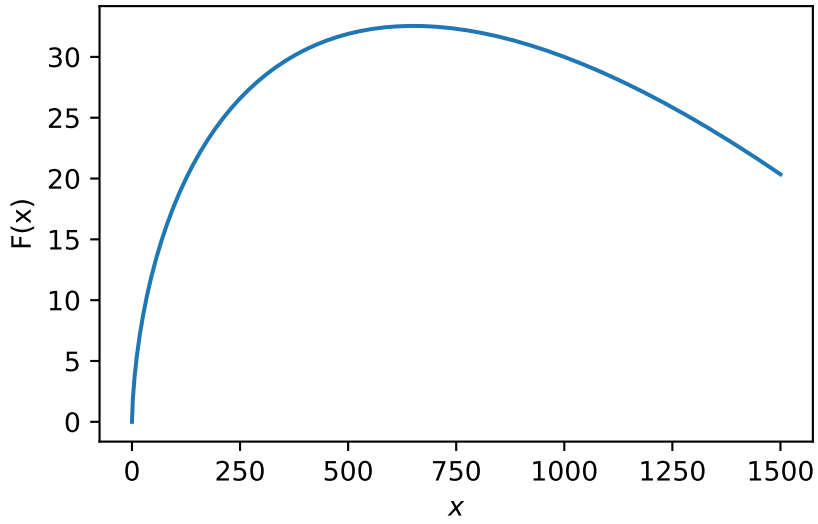


Figure 5.1: CNT style free energy landscape given by $F(x) = -\Delta\mu x + \phi\gamma x^{\frac{2}{3}}$ with $\Delta\mu = 0.1$ and $\phi\gamma = 1.3$.

which we shall refer to as the linear and reciprocal length forms respectively. Here ξ is a constant and N_M is the memory length. In the linear form, the longer the memory a system has, the greater its possible impact on the dynamics $\max(\mathcal{M}_L(m)) = \xi N_M$. Whereas in the reciprocal length form the memory has the same maximum effect regardless of the memory length, $\max(\mathcal{M}_R(m)) = \xi$, in this case however the longer the memory length the finer the incremental effect the memory has on the system.

We have discussed how we generate our reference stochastic processes in terms of probabilities of attempting and accepting moves, however to conduct the likelihood analysis we need the probabilities with which the moves are made. Specifically we require three probabilities, the probability of moving in the positive direction $P(x+1|x,m)$, the probability of moving in the negative direction $P(x-1|x,m)$, and the probability of making no move $P(x|x),m$. The first two of these are simply the product of attempting and accepting a move in the given direction,

$$P(x+1|x,m) = P_{\text{Attempt}}(+|m)P_{\text{Accept}}(x+1|x,m) \tag{5.13}$$

$$P(x-1|x,m) = P_{\text{Attempt}}(-|m)P_{\text{Accept}}(x-1|x,m). \tag{5.14}$$

The probability of making no move is the sum of the probability of moving in either direction and rejecting that move,

$$P(x|x,m) = P_{\text{Attempt}}(+|m)\left[1 - P_{\text{Accept}}(x+1|x,m)\right] + P_{\text{Attempt}}(-|m)\left[P_{\text{Accept}}(x-1|x,m)\right]. \quad (5.15)$$

It is worth noting that the attempt and acceptance probabilities are individually bounded between 0 and 1. This has no effect in the memoryless case, $N_M = 0$, however does have an effect in for systems with memory. To save calculating these probabilities repeatedly we construct a transition matrix

$$T_{x_2 x_1 m} = P(x_2|x_1, m), \quad (5.16)$$

for each energy landscape, memory length and memory function.

The transition matrix for the reference stochastic processes will be tri-diagonal because as the coordinate can only move in steps of size 1, $P(x_2|x_1, m) = 0 \quad \forall \quad |x_2 - x_1| > 1$. Probabilities of 0 present a practical problem for our likelihood analysis as $\ln(0) \quad \to -\infty$ which will cause problems for our numerical computation if we encounter steps of a size greater than 1. While steps of this size are not possible in our reference stochastic processes they are likely to occur in other systems. To combat this we take the minimum value of each element of the transition matrix to be some small value, $\min(T_{x_2 x_1 m}) = 2.2250738585072014 \times 10^{-308}$. This particular value is chosen for practical reasons, it is a constant in the Python numpy package which we use to conduct our analysis, `numpy.finfo(numpy.float64).tiny`.

5.1.2 Toy Model Results

To evaluate this likelihood method we generate ensembles of 10 trajectories for a range of memory lengths (0-10). We will refer to the memory length used to generate a given trajectory as the "trajectory" memory length $N_{M,\text{traj}}$. These ensembles are then evaluated using the average likelihood approach with different assumed memory lengths. We will refer to the memory length assumed in the likelihood analysis as the "scoring" memory length $N_{M,\text{score}}$. If the likelihood approach is valid then we expect to find the ensemble with the highest likelihood, $\ell_{N_{M,\text{score}}}$, corresponds to $N_{M,\text{score}} = N_{M,\text{traj}}$.

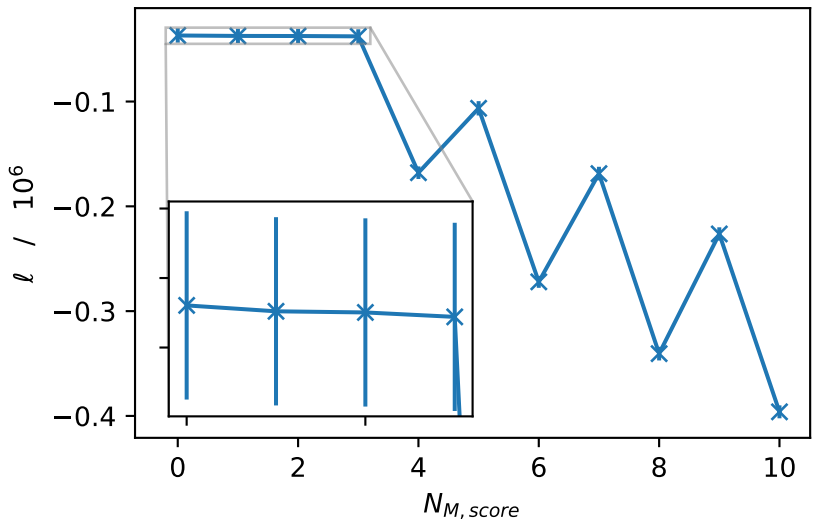


Figure 5.2: Ensemble likelihood scores ℓ for a memoryless random walk on a flat free energy landscape, scored as the likelihood that the ensemble of walks is a realisation of a reference stochastic process with a memory length $N_{M,\text{score}}$ and a linear memory function $\mathcal{M}_L$.

Linear Memory Function

Figure 5.2 shows the likelihood score ℓ for a memoryless random walk if we use a scoring memory, $N_{M,\text{score}}$ of 0 to 10. If we just consider the mean score for the ensembles with each memory length, the scoring evaluates a memory length of 0 as the most likely for the ensemble. However, when considering the errors we cannot distinguish memory lengths of 0 to 3 from each other. We can however determine that the ensemble is most likely to have a memory length in the range $(0, 3)$ or in other words we can say that the results are indistinguishable from a memoryless random walk on a flat energy landscape. While increasing the number of trajectories per ensemble may reduce the standard error, and so make the results distinguishable, we choose 10 trajectories per ensemble as it is a number which is realistically accessible from seeded nucleation runs.

It is worth noting that the memory function, $\mathcal{M}_L(m)$, used to both generate and score the trajectories has a value of $\xi = 0.250001$ (the 0.000001 is present to avoid dividing by zero) meaning that the point at which the score becomes distin-

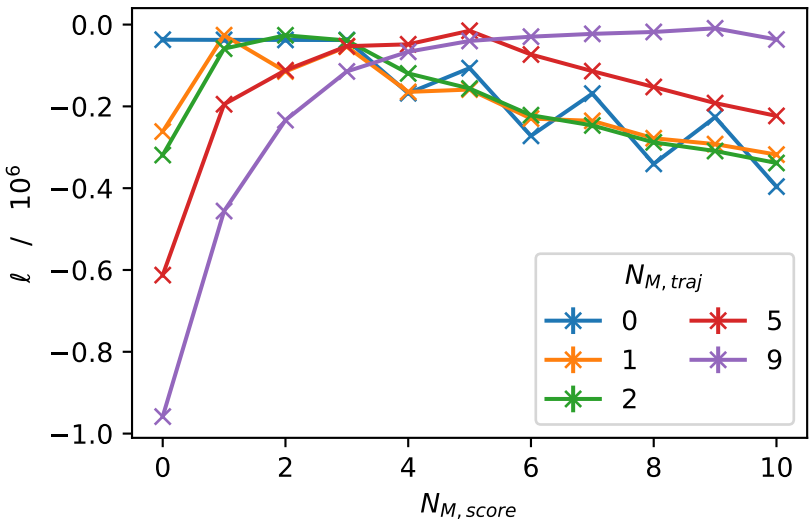


Figure 5.3: Ensemble likelihood scores ℓ for random walks on a flat free energy landscape with a range of memory lengths $N_{M,\text{traj}}$, each scored as the likelihood that the ensemble of walks is a realisation of a reference stochastic process with a memory length $N_{M,\text{score}}$ and a linear memory function $\mathcal{M}_L$.

guishably lower coincides with $\mathcal{M}_L(m) > 1$. This drop off happens at $\mathcal{M}_L(m) > 1$ for all values of ξ tested on flat energy landscapes.

The likelihood scores for ensembles with none-zero memory lengths on flat energy landscapes are shown in figure 5.3. For all of the none-zero memory lengths the scoring correctly identifies the memory length used to generate the trajectories with larger memory lengths being separated by a larger number of standard errors. The non-zero memory length with the closest "incorrect" value is $N_{M,\text{traj}} = 2$ for which the score for $N_{M,\text{score}} = 3$ is within 2 standard errors of, but lower than, the score for $N_{M,\text{score}} = 2$.

Figures 5.4 and 5.5 compare the scores for trajectories on a quadratic and CNT style potentials respectively. For the quadratic potential the scoring correctly identifies the trajectory memory for all memory lengths, although some of the neighbouring points are within a few standard errors. Notably the scores at longer memory lengths have larger standard errors which make it harder to distinguish between systems with longer memories and also means the scores for these memory lengths are within fewer standard errors of "correct" values for shorter memory trajectories.

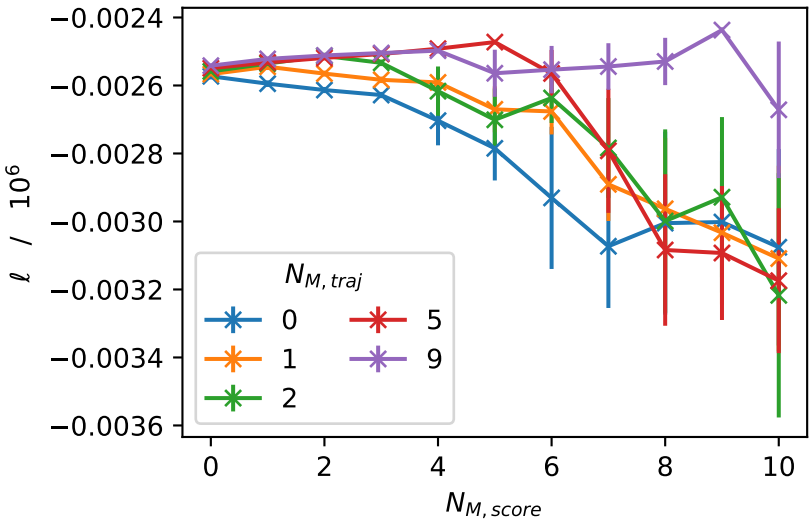


Figure 5.4: Ensemble likelihood scores ℓ for random walks on a quadratic free energy landscape with a range of memory lengths $N_{M,\mathrm{traj}}$, each scored as the likelihood that the ensemble of walks is a realisation of a reference stochastic process with a memory length $N_{M,\mathrm{score}}$ and a linear memory function $\mathcal{M}_L$ on the same energy landscape.

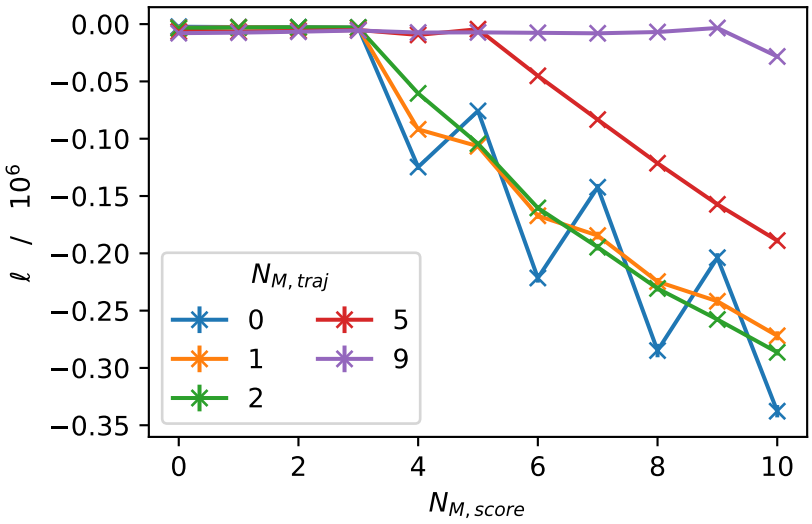


Figure 5.5: Ensemble likelihood scores ℓ for random walks on a CNT style free energy landscape ($F(x) = \phi\gamma x^{2/3} - \Delta\mu x$) with a range of memory lengths $N_{M,\text{traj}}$, each scored as the likelihood that the ensemble of walks is a realisation of a reference stochastic process with a memory length $N_{M,\text{score}}$ and a linear memory function $\mathcal{M}_L$ on the same energy landscape.

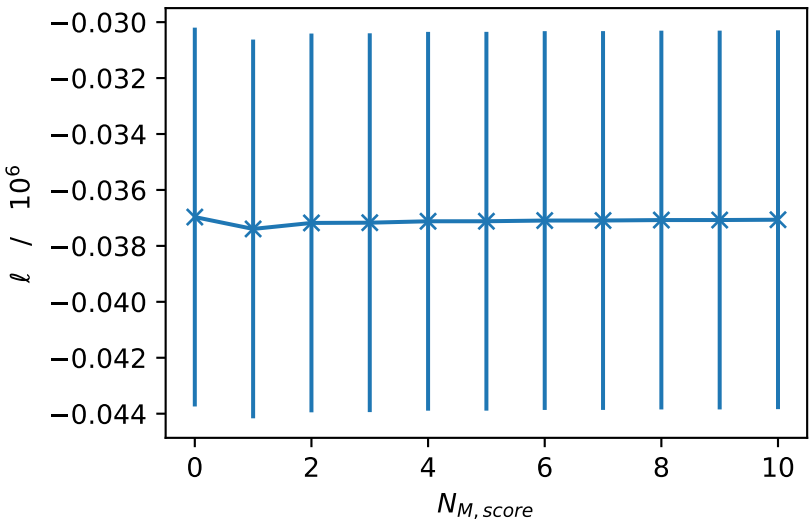


Figure 5.6: Ensemble likelihood scores ℓ for a memoryless random walk on a flat free energy landscape, scored as the likelihood that the ensemble of walks is a realisation of a reference stochastic process with a memory length $N_{M,\text{score}}$ and a reciprocal length memory function $\mathcal{M}_R$.

For the CNT style potential the scoring again correctly identifies the memory length for the trajectories. As with the quadratic potential some of the other points are within 2 standard errors of the "correct" value. This time this point are associated with the mid to long memory lengths whereas the scores of the shorter memory lengths are more standard errors apart, despite their closer absolute proximity.

Reciprocal Length Memory Function

All of the predictions so far are using the "linear" memory function $\mathcal{M}_L(m)$. Figures 5.6, 5.7, 5.8 and 5.9 show the results when the "reciprocal length" variant, $\mathcal{M}_R(m)$, is used to both generate the trajectories and score the results.

As in the case of $\mathcal{M}_L(m)$, despite the ensemble average values correctly identifying the memoryless trajectory on the flat landscape, it is indistinguishable from the scores of the other memory lengths when considering the standard errors, figure 5.6. The most obvious difference between the memory functions $\mathcal{M}_L(m)$ and $\mathcal{M}_R(m)$ is that there appears to be no drop off point for the reciprocal length

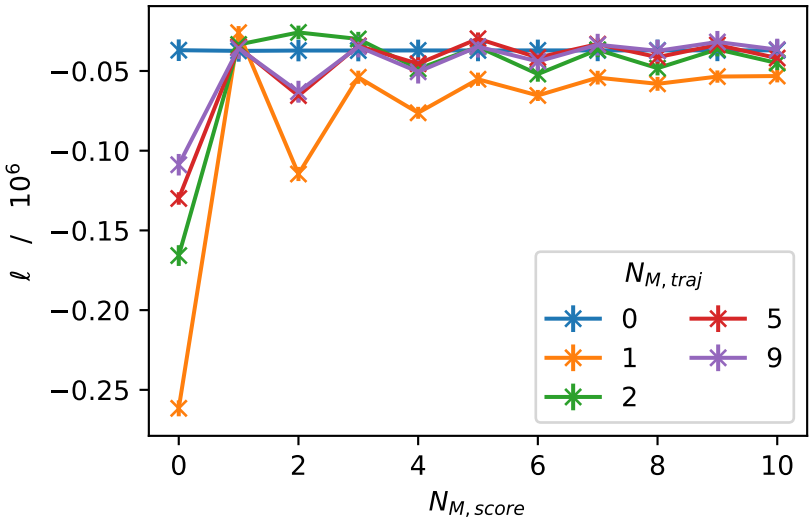


Figure 5.7: Ensemble likelihood scores ℓ for random walks on a flat free energy landscape with a range of memory lengths $N_{M,\text{traj}}$, each scored as the likelihood that the ensemble of walks is a realisation of a reference stochastic process with a memory length $N_{M,\text{score}}$ and a reciprocal length memory function $\mathcal{M}_R$.

memory function. This is possibly related to the conversion of the condition for the linear memory function $\mathcal{M}_L(m) > 1$ to the reciprocal memory function. This might convert to $\mathcal{M}_R(m) = \xi m/N_M > 1$ or, for the case of maximal memory influence $m = N_M$, $\xi > 1$ which would mean the condition is not met for these trajectories with $\xi = 0.25$.

For the flat energy landscape, figure 5.7, the scores identify the correct energy lengths. However, as with the linear memory function, the errors make it hard to distinguish between nearby points. With the reciprocal length memory function the trajectories with longer memories are harder to distinguish amongst themselves than those with shorter memories, which is the opposite of the linear memory function trajectories.

The scores for the quadratic energy landscape correctly identify the trajectories memory, but again for most of the ensembles the standard error makes several of the points effectively indistinguishable. For example, from figure 5.8 we could reasonably identify the orange line as having a memory of 1, which is the expected result. We may be comfortable identifying the green points as having a memory

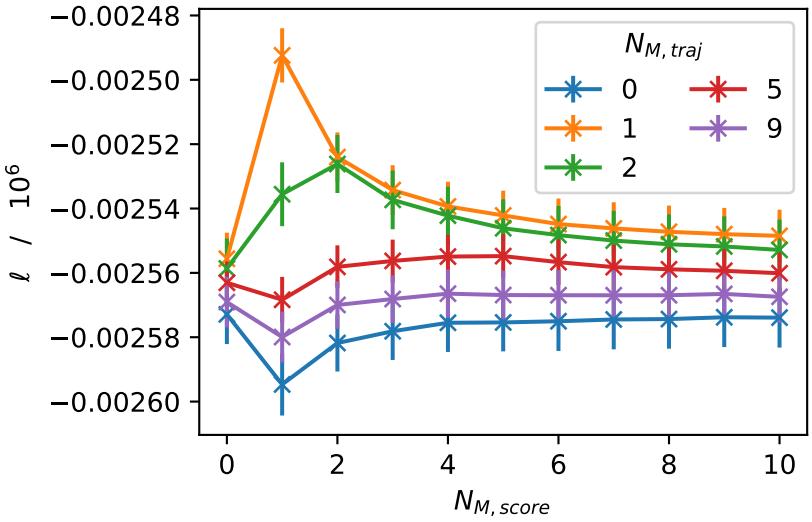


Figure 5.8: Ensemble likelihood scores ℓ for random walks on a quadratic free energy landscape with a range of memory lengths $N_{M,\text{traj}}$, each scored as the likelihood that the ensemble of walks is a realisation of a reference stochastic process with a memory length $N_{M,\text{score}}$ and a reciprocal length memory function $\mathcal{M}_R$ on the same energy landscape.

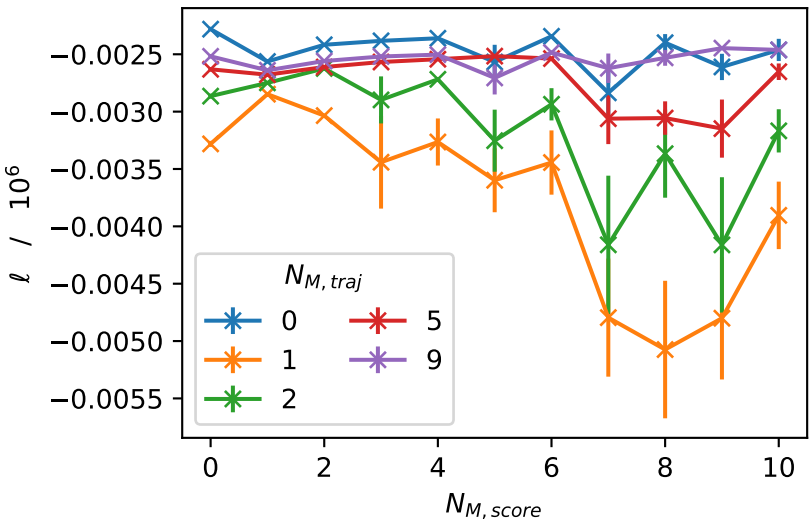


Figure 5.9: Ensemble likelihood scores ℓ for random walks on a CNT style free energy landscape ($F(x) = \phi\gamma x^{2/3} - \Delta\mu x$) with a range of memory lengths $N_{M,\text{traj}}$, each scored as the likelihood that the ensemble of walks is a realisation of a reference stochastic process with a memory length $N_{M,\text{score}}$ and a reciprocal length memory function $\mathcal{M}_R$ on the same energy landscape. Here $\Delta\mu = 0.1$ and $\phi\gamma = 1.3$.

of around 2. However we cannot strongly identify any of the other lines plotted as being significantly different from the memoryless case. On this landscape the reciprocal length memory function is harder to identify than the linear function. This difference might be expected if we consider how the different functions operate and the consequences of a quadratic potential. In a quadratic potential the motion of the walker is heavily confined, it takes increasingly lower probability steps to move away from the minima. This means that none of the trajectories are likely to have a large value of m, despite increasing N_M. For the systems with a linear memory function $\mathcal{M}_L(m)$ the influence of each increment in m has a weight of ξ, however for the reciprocal length function $\mathcal{M}_R(m)$ each increments only has a influence of ξ/N_m meaning that the memory in systems with larger memory lengths have less influence per increment in m. This explains both why the longer memory length trajectories become closer to the memoryless scores, their memory is having less of an effect, and why trajectories with memory of length 1 and 2 are easier to identify, their memory is having the largest effect on the dynamics.

Figure 5.9 shows the reciprocal length memory function results for the CNT style energy landscape. Once again the score correctly identifies the memory of the ensembles, with the standard errors giving rise at ambiguity between some of the points. These trajectories all start at $x_0 = 350$, which is lower than the critical value but still in the flatter, less constrained region of the landscape.

In all but the most extreme cases this scoring analysis identifies the memory present in a system to some degree. It is obvious that the uncertainty in some cases complicates this identification. Because of this this process may not be able to determine the exact amount of memory in a more complex system, but it may be able to distinguish between systems with long, short and no memory. It is also worth noting that we are only using this analysis to identify systems with memory in the two forms we have discussed, linear and reciprocal length, It is entirely possible, if not probable, that the memory mechanics are more complex or just different in other systems and so we may struggle to detect it.

5.2 Ising model

We now apply this same methodology to the dynamics of nucleation during magnetisation reversal in the 2D Ising model. Specifically, we will generate nucleation trajectories and compute the likelihood that these can be described by a memoryless random walk in the reaction coordinate. In order to do this we first need an expression for the free energy of the system as a function of largest cluster size, our chosen reaction coordinate to construct a transition matrix for likelihood scoring calculations. For this expression we fit free energies calculated using umbrella sampling and fit them using equation 3.13 described in section 3.2. These umbrella sampling results and fittings where carried out by another member of the group, Dr Dipanjan Mandal, based on work by Ryu and Cai [37]. These free energy functions are shown in figure 5.10 and the parameters of the fits are tabulated in table 5.1. For our Ising model simulations we will use the parameters $T = 1.2$ and $h = 0.1$, with a grid size of 64×64.

5.2.1 Diffusion Events

In our simple 1D models, because of the model setup, every simulation step was a diffusion event. In the Ising model this is no longer strictly the case. As we want

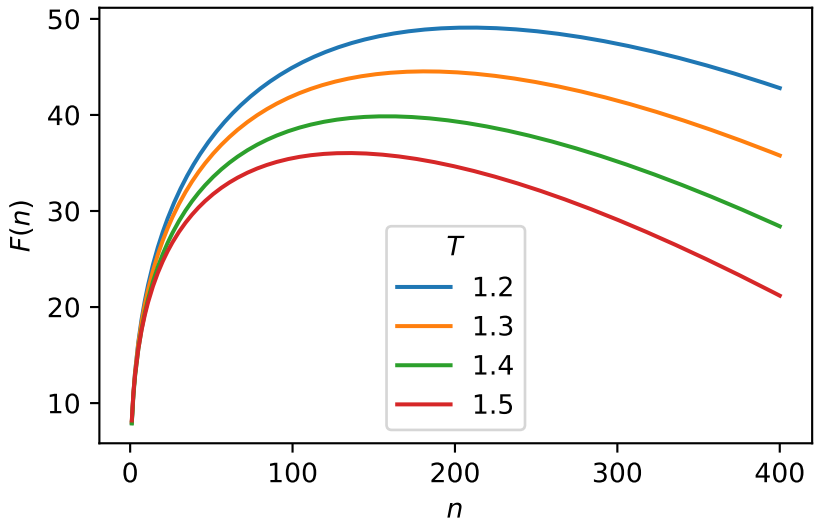


Figure 5.10: Free energy landscape for 2D Ising model at various temperatures T.

h	T	c	a_1	a_2
0.1	1.2	2.55	5.59	1.41
0.1	1.3	3.05	5.16	1.59
0.1	1.4	3.30	4.77	1.62
0.1	1.5	4.06	4.32	1.79

Table 5.1: Parameters for free energy of 2D Ising model in the form $F(n) = c - 2hn + a_1 n^{\frac{1}{2}} + a_2 \ln(n)$.

to assess how well this system is described by a diffusive random walk of the free energy landscape, we need to determine the timescale over which the diffusion of the reaction coordinate occurs for this system. To do this, we run an ensemble of simulations starting at the top of the free energy barrier, we then use equation 4.6 to calculate the diffusion coefficient, D, as described in section 4.1.2. For the Metropolis spin flip dynamics described in section 2.1.2, results of these calculations give $D = (0.005244 \pm 0.000001)$ simulation steps^{-1} and therefore $\tau_D = 1/D = (191 \pm 1)$ simulation steps, to the closest integer step. Here a simulation step is a single attempt to flip the spin at a single site, not the more common Monte Carlo sweep in which we perform 1 flip attempt per grid site, though we do not necessarily attempt to flip every site.

5.2.2 Single Step Size

In the toy 1D models each step was constrained to be size 1, 0 or -1, however in Ising model simulations, this constant no longer applies. When we apply the scoring methodology in the same as for the random walk models, any moves with a size greater than 1 will have a probability of 0 (or as discussed earlier some small value). This may be the desired effect, but to explore what happens if we allow then we must alter our scoring process to account for them. One simple method of allowing move sizes larger than one is, phrased in terms of 1D random walks, have the system choose to go left or right and then choose the size of step it will make in that direction. To implement this, we need the probability of making different move sizes, which for simplicity we take as the distribution of moves made by our Ising model simulations, shown in figure 5.11.

5.2.3 Ising Model Results

To begin, we will look at the score for systems where we ignore the possibility of step sizes greater than 1. Figures 5.12 and 5.13 show the scores for various starting points, n_0, assuming a linear and reciprocal length memory function respectively. For the linear memory function the smallest starting size, $n_0 = 69$ identifies the system as memoryless, however this starting point is in a region of the energy landscape that is highly constrained. As we saw with the quadratic potential in section 5.1.2 this can make the scoring unreliable, however it should not be ignored. The other 4 starting points are less conclusive, but indicate a memoryless or low memory process. The

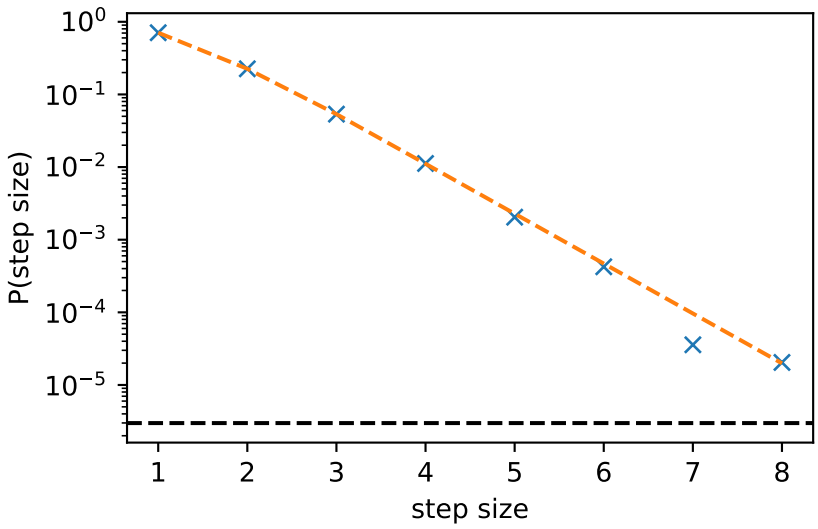


Figure 5.11: Probability of making a step of the nucleus size coordinate of any size during our Ising model simulations. ×s show the probability observed in the simulation results and the orange dashed line is a fit of those points. The fit is of the form $P(s) = A\exp\{Bs^2\} + C\exp\{Ds\}$, with parameters $A = -0.827$, $B = -0.911$, $C = 3.628$ and $D = -1.582$. The black dashed line indicates the lowest observable probability in the sample $1/N_{\text{samples}} = 1/334720$.

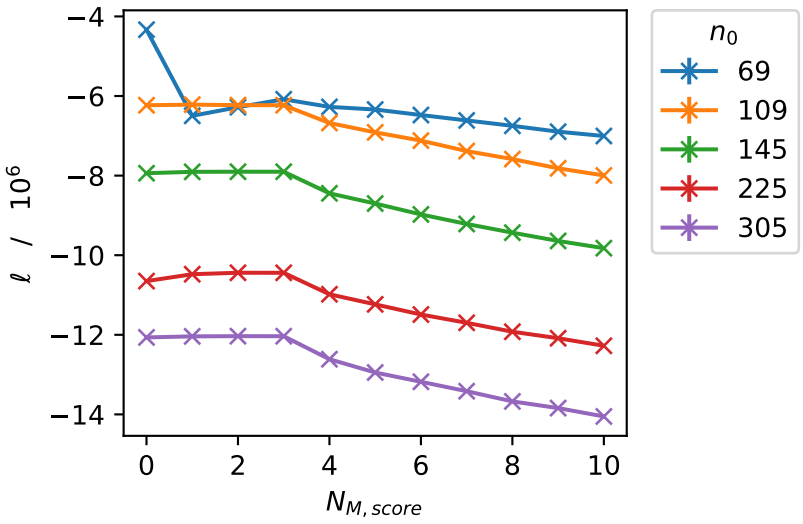


Figure 5.12: Ensemble likelihood scores ℓ for 2D Ising model trajectories with various starting cluster sizes n_0. The trajectories are scored as the likelihood that they are represented by a 1D random walk with fixed steps of size 1 on a free energy landscape of the form $F(n) = c - 2hn + a_1 n^{\frac{1}{2}} + a_2 \ln(n)$ with a linear memory function $\mathcal{M}_L$ and memory lengths $N_{M,\text{score}}$.

$n_0 = 225$ starting point has a slightly lower score for a memoryless system than one with a short memory, although the difference is only around 2%. Given these results we conclude that if the 2D Ising model is described by a random walk on its free energy landscape with linear memory function, it is best described by as memoryless or with a short memory at most.

The scores using a reciprocal memory function, figure 5.13, tell a similar story to the linear memory function. The smaller 2 starting positions indicate a system with a short or no memory, while the other starting points are inconclusive. Once again the $n_0 = 225$ starting points suggests a non-zero memory however the margin is small. Both the linear and reciprocal length scores are most similar to those of the memoryless random walk scores from section 5.1.2 and we conclude that they show at most a short memory.

Now we have examined the results for a diffusion step size of 1, we can look at the scores for which we allowed steps of sizes greater than 1. Figure 5.14 shows the results for a linear memory function and indicates a short memory length. The score

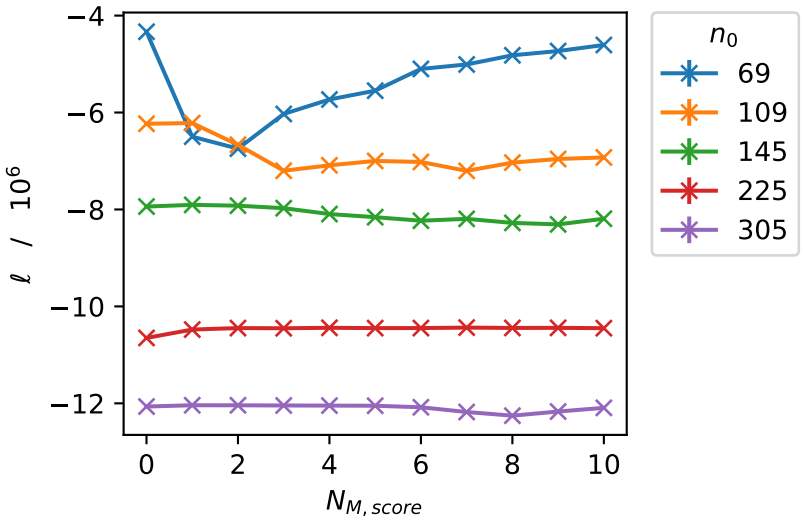


Figure 5.13: Ensemble likelihood scores ℓ for 2D Ising model trajectories with various starting cluster sizes n_0. The trajectories are scored as the likelihood that they are represented by a 1D random walk with fixed steps of size 1 on a free energy landscape of the form $F(n) = c - 2hn + a_1 n^{\frac{1}{2}} + a_2 \ln(n)$ with a reciprocal length memory function $\mathcal{M}_R$ and memory lengths $N_{M,\text{score}}$.

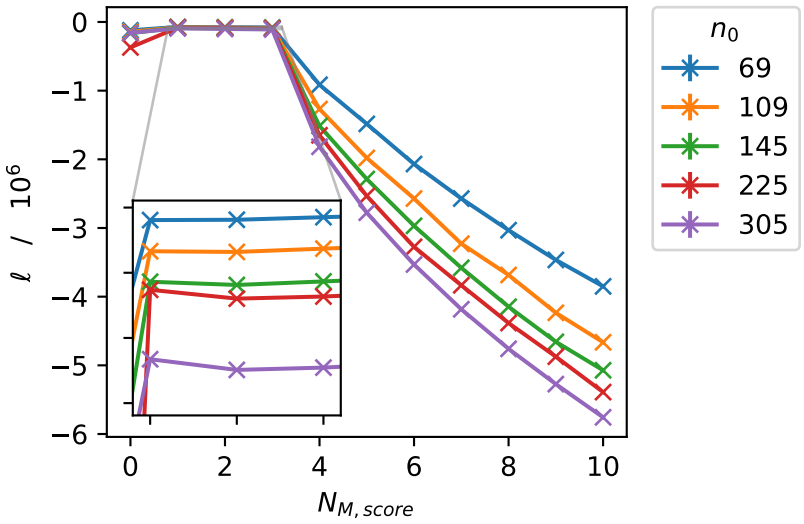


Figure 5.14: Ensemble likelihood scores ℓ for 2D Ising model trajectories with various starting cluster sizes n_0. The trajectories are scored as the likelihood that they are represented by a 1D random walk with a variable step size on a free energy landscape of the form $F(n) = c - 2hn + a_1 n^{\frac{1}{2}} + a_2 \ln(n)$ with a linear memory function $\mathcal{M}_L$ and memory lengths $N_{M,\text{score}}$.

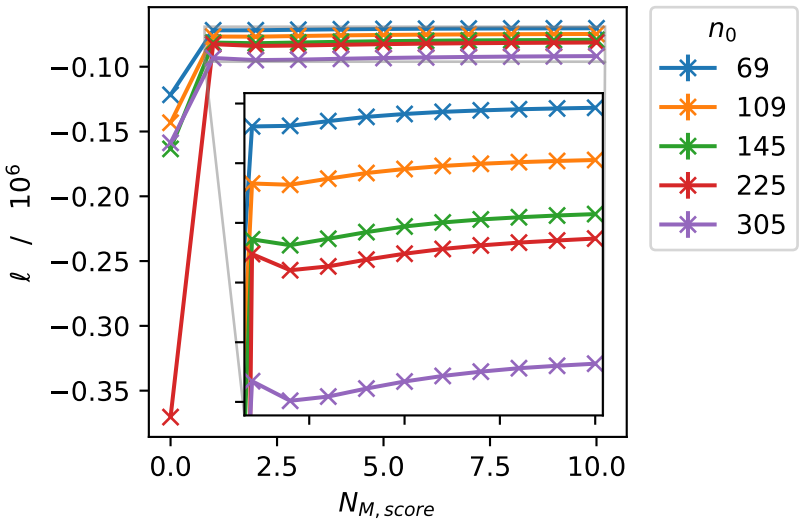


Figure 5.15: Ensemble likelihood scores ℓ for 2D Ising model trajectories with various starting cluster sizes n_0. The trajectories are scored as the likelihood that they are represented by a 1D random walk with a variable step size on a free energy landscape of the form $F(n) = c - 2hn + a_1 n^{\frac{1}{2}} + a_2 \ln(n)$ with a reciprocal length memory function $\mathcal{M}_R$ and memory lengths $N_{M,\text{score}}$.

for the memoryless is noticeably less than for memory lengths 1,2 and 3 suggesting a short memory is more likely than no memory.

The score for the reciprocal length memory function are show in figure 5.15. For this case, memory lengths ≥ 1 score similarly however a memory length of zero score substantially lower for all starting points. Both the linear and reciprocal length memory functions suggest that if we allow for diffusion steps with sizes larger than 1 in this way our trajectories are best described with memory. We cannot be certain if these effects are because of memory in the model which generated the trajectories or because of the method we have used to score those trajectories. All we can assess is if the dynamics of the trajectories can be described in this way, what is the most likely memory length for those dynamics. Also, this simplistic modelling of step sizes does not have any dependence on the current value of n, meaning all steps are equally likely at all points on the free energy landscape, which may not be the case.

5.3 Molecular Dynamics

We now apply the scoring analysis to molecular dynamics trajectories. We use the Lennard-Jones system described in section 2.3.2, with the only modification being that we report the largest cluster size every simulation step, rather than every 1000 steps. For these simulations we set a temperature of $T = 0.53$, and set all other system parameters the same as chapter 4. By using these simulation parameters we can use the free energy landscape generated in the single temperature seeding method, section 4.1.3. Before we can score the trajectories we first have to sub-sample them to our diffusion event timescale. The diffusion coefficient for this system id $D = 116$, this is calculated in section 4.1.2, giving a diffusion timescale for the largest cluster reaction coordinate n of $\tau_D = 0.0086t^*$, where t^* is the Lennard-Jones time unit. We can now sub-sample our trajectories and continue with the scoring.

Figures 5.16 and 5.17 show the scoring for these MD trajectories with linear and reciprocal length memory functions respectively. Both sets of scores are most similar to the memoryless random walks with their respective memory functions, with the linear memory function giving similar scores for short memories and the reciprocal length memory function giving similar scores for all memory lengths.

As in the Ising model results, some of the move sizes are greater than 1. As we described in section 5.2.2, we can use the trajectories to calculate the probability of each move size and account for that in the scoring process. The distribution of moves for the MD trajectories is shown in figure 5.18.

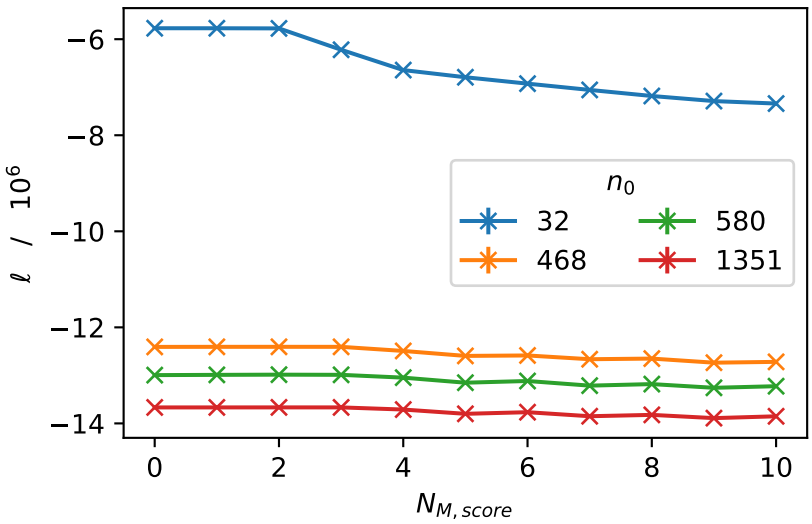


Figure 5.16: Ensemble likelihood scores ℓ for LJ MD trajectories with various starting cluster sizes n_0. The trajectories are scored as the likelihood that they are represented a random walk with a fixed step size of 1 on the CNT free energy landscape $F(n) = -n|\Delta\mu| + \phi\gamma n^{2/3}$) with a linear memory function $\mathcal{M}_L$, and a memory length of $N_{M,\text{score}}$. Here the temperature for the simulation is $T = 0.534$.

Similarly to the result for the Ising model, the scores for the MD trajectories allowing variable length move indicate a system with some non-zero memory. For the linear memory function, figure 5.19 suggests a short (but non-zero) memory, where as the reciprocal length memory function, figure 5.20, indicates that the system has a non-zero memory, but with only a slightly higher score for shorter memories.

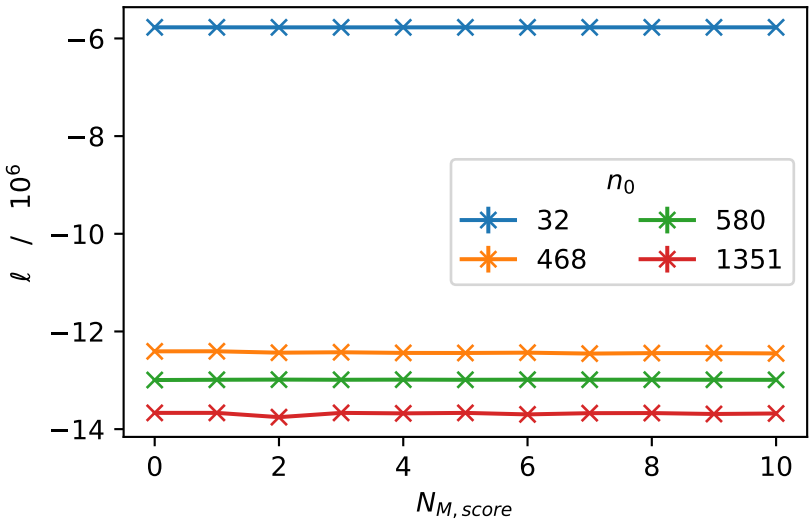


Figure 5.17: Ensemble likelihood scores ℓ for LJ MD trajectories with various starting cluster sizes n_0. The trajectories are scored as the likelihood that they are represented a random walk with a fixed step size of 1 on the CNT free energy landscape $F(n) = -n|\Delta\mu| + \phi\gamma n^{2/3}$) with a reciprocal length memory function $\mathcal{M}_R$, and a memory length of $N_{M,\text{score}}$. Here the temperature for the simulation is $T = 0.534$.

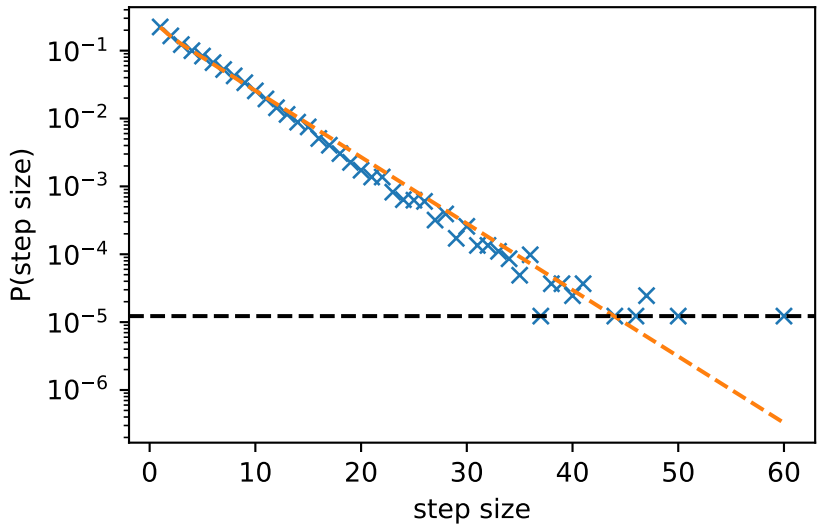


Figure 5.18: Probability of making a step of any size during our MD simulations. ×s show the probability observed in the simulation results and the orange dashed line is a fit of those points. The fit is of the form $P(s) = A\exp\{Bs^2\}$, with parameters $A = 0.275$ and $B = -0.244$. The black dashed line indicates the lowest observable probability in the sample $1/N_{\text{samples}} = 1/81540$.

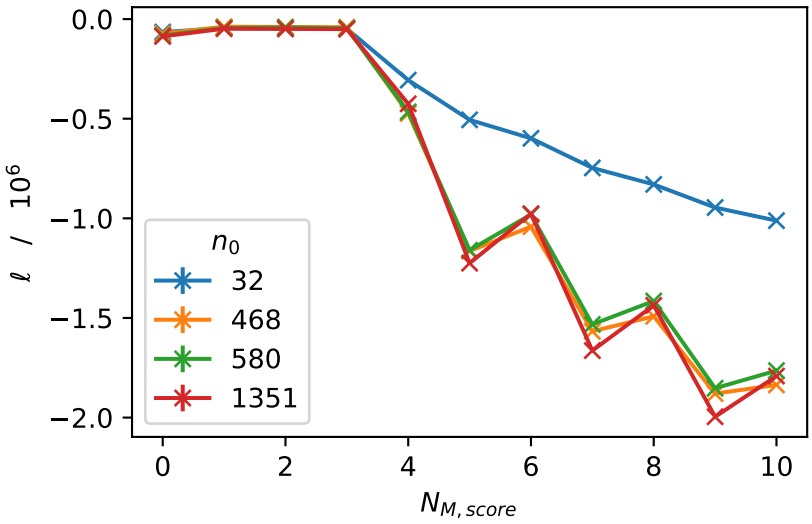


Figure 5.19: Ensemble likelihood scores ℓ for LJ MD trajectories with various starting cluster sizes n_0. The trajectories are scored as the likelihood that they are represented a random walk with a variable step size on the CNT free energy landscape $F(n) = -n|\Delta\mu| + \phi\gamma n^{2/3})$ with a linear memory function $\mathcal{M}_L$, and a memory length of $N_{M,\text{score}}$. Here the temperature for the simulation is $T = 0.534$.

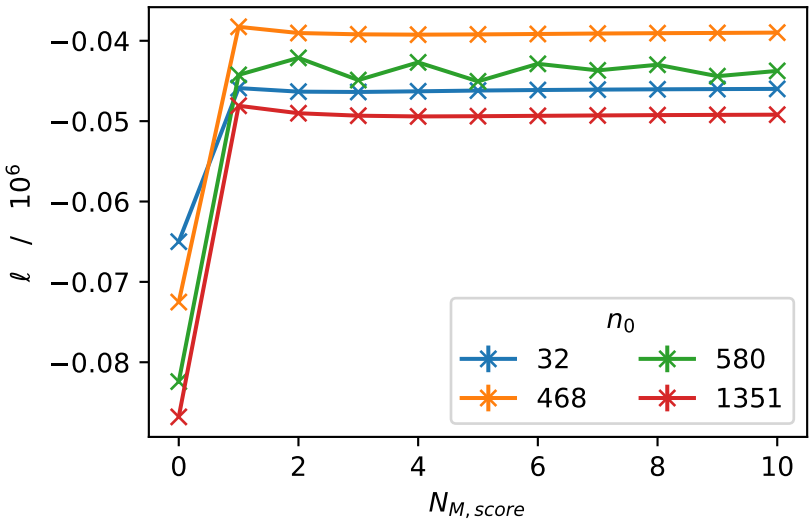


Figure 5.20: Ensemble likelihood scores ℓ for LJ MD trajectories with various starting cluster sizes n_0. The trajectories are scored as the likelihood that they are represented a random walk with a variable step size on the CNT free energy landscape $F(n) = -n|\Delta\mu| + \phi\gamma n^{2/3}$) with a reciprocal length memory function $\mathcal{M}_R$, and a memory length of $N_{M,\text{score}}$. Here the temperature for the simulation is $T = 0.534$.

5.4 Conclusion

By assuming that the dynamics of nucleation can be described as stochastic dynamics on an energy surface, we demonstrate a method of identifying the strength-/length of memory in these dynamics, given some memory function. We considered two memory functions, a linear memory function $\mathcal{M}_L(m)$ with a contribution proportional to ξm, and a reciprocal length function $\mathcal{M}_R(m)$ with a contribution proportional to $\xi m/N_M$, where m is the current memory state, N_M is the total memory length, and ξ is a constant determining the strength of the memory's effect.

We tested this method on 1D random walks which we implemented with known memories, finding that while the correct memory length was recovered for all cases when considering the absolute score, when considering the errors on these scores it would be hard to practically separate some of them. We also found that the errors on our scores, and so our ability to distinguish between them, was dependent on the form of the energy landscape. This makes it harder to quantify memory in highly constrained potentials, for example the quadratic potential in our test cases or small cluster sizes in the CNT energy landscape.

After verifying the method on simple test cases, we applied it to the 2D Ising model, using it to score trajectories against a free energy landscape constructed from umbrella sampling. Instead of recording the largest cluster size every sweep (or multiple sweeps) as is usual, we calculated it every attempted flip. The purpose of this more frequent cluster calculation was to allow us to sub-sample the trajectories down to the diffusion of the nuclear size coordinate timescale which was calculated using the same methodology as in the CNT seeding methods, section 4.1.2, to be $\tau_D = 191$ steps. While scoring these trajectories we found that many of the moves were larger than 1, which is not the case for the 1D random walks previously considered. These move sizes could be the effect of, or amplified by, an incorrect measurement of τ_D, or just a part of the system as one spin flip could join two clusters of arbitrary size in a single step. This also points to the largest cluster size not being the ideal reaction coordinate for the 2D Ising model with spin flip dynamics. Whatever the cause of these larger diffusion steps, we determine the probability distribution of each move size present in our trajectories and include this in our scoring methodology.

We scored the Ising model trajectories using both of our memory functions both ignoring and accounting for diffusion steps greater than 1. When we ignored the larger diffusion steps, scoring them with a low likelihood, both of the memory

functions identified the system as either being memoryless or having a short memory. When accounting for a variable move size the linear memory function indicates a short but non-zero memory length whereas the reciprocal length memory function only indicates a memoryless system being least likely. Comparing the likelihoods ignoring and accounting for steps sizes greater than 1 we find that the most likely description of the trajectories is either a short but non-zero memory with a linear memory function allowing for step sizes > 1, or a reciprocal memory function with a long memory length again allowing for step sizes > 1.

When applying this same analysis to trajectories generated using MD, we again calculate the size of largest cluster frequently and sub sample down to the diffusion timescale, in this case $\tau_D = 0.0086t^*$ (43 simulation steps). We again choose to conduct the analysis twice, first ignoring step sizes greater than 1, and then accounting for them in the same way as before. When ignoring larger steps the linear memory function indicates the system is memoryless or has a short memory whereas the reciprocal length memory function is unable to distinguish between any memory length. When accounting for a variable step size both memory functions indicate a non-zero memory length with both favouring a short memory, but only by a small margin in the case of the reciprocal length memory function. The likelihood analysis suggests that the best representation of these trajectories is a non zero memory length accounting for step sizes > 1, with the linear memory function indicating a short memory, and the reciprocal length memory function only being able to determine a memory length > 0.

The results for both the Ising model and MD trajectories favour a non-zero memory length accounting for step sizes > 1, and are equally well represented by both the linear and reciprocal length memory functions. In both cases with the linear memory function a short memory length is most likely. Whereas, for the reciprocal length memory function the Ising model is best represented by a long memory length, and for the MD trajectories all memory lengths > 0 are similarly likely with perhaps a slight favour for short memories. However we can only say how relatively likely each of the memory lengths are assuming that the trajectories are well described by random walks on the given energy landscapes with our chosen memory functions. Other memory functions could be defined which could lead to different results.

When generating the trajectories to score, we report the cluster size at a high rate and then sub-sample down to the diffusion timescale. While this is not overly

arduous for small systems with simple reaction coordinates, it is wasted computational effort which could become a big factor in larger or more complex systems. It may be possible to, instead of sub-sampling the trajectories, construct the transition matrix to account for multiple diffusion events. This way the transition matrix could be built for the output frequency of the simulation rather than wasting compute time to ensure that the frequency is high enough to sub-sample to the diffusion timescale. While the method of extending the transition matrix for the memoryless case is simple, $T_{2\text{steps}} = T_{1\text{step}}^2$, it is more complicated for non-zero memories. One approach would be to break down the transition matrix into one for each memory value, extend them individually and recombine them. This would make it necessary to have the number of diffusion steps be a multiple of the memory length as is is not possible to track a path of smaller increments through the transition matrix. Whatever the method of constructing these transition matrices, they would only be useful up to a limit as the errors in the diffusion timescale and matrix finite size effect will compound to reduce their accuracy and ability to distinguish trajectories.

The two memory functions we compared were chosen to represent a system where the memory effect gets stronger as it can remember more (the linear memory function), and a system in which memory has a finite effect regardless of length (reciprocal length memory function). Both of these functions can be represented by $m \times$ constant. With the linear memory function we can see the effect of varying the memory length while leaving the constant unchanged, however with the reciprocal length memory function the memory length and this constant are linked. If we take a single memory length we could vary ξ in order to determine its effect at each memory length which may allow us to optimise our reference stochastic processes to give the highest likelihood.

These results are sightly unexpected. As we discussed in section 3.2, other work shows the Ising model with spin flip dynamics, under magnetisation reversal, to be well described my Markovian dynamics. Also, the results from chapter 4 indicate that the seeded MD trajectories conform to the CNT assumptions which suggests that they too are well described by Markovian dynamics. However the results from this chapter would indicate that there is more to the picture.

An obvious avenue for future work is to experiment with different memory functions, possibly weighting more recent events stronger that older ones. It would also be interesting to compare trajectories from the Ising model with spin flip and spin exchange dynamics. Comparing these two sets of dynamics would allow for a

similar comparison to that of Kupiters and Barkema [36] discussed in section 3.2. It may also be possible to find analytic solutions for simple energy landscapes with simple memory functions.

The non-Markovian behaviour in these dynamics may be due to the size of largest cluster not being an ideal reaction coordinate, and that the free energy barrier may be more accurately described by a different reaction coordinate, or a collection of multiple reaction coordinates. Another avenue of further work could be to take a system with a free energy landscape known to be, or designed to be, well described by combination of parameters, but comparatively poorly by a single one of those parameters. Memory could then be added to random walks based on a single parameter to see if it would better represent the higher dimensional dynamics.

Chapter 6

Committor Prediction

While the cluster size n is a physically motivated choice of reaction coordinate for CNT calculations, as discussed in 2.5, it is not the ideal choice of reaction coordinate. As we discussed in 2.5.2 the ideal choice of reaction coordinate is the committor, p_B, which is defined for each microstate as the probability of trajectories which visit that microstate reaching the product state **B** before returning to reactant state **A**. For nucleation the reactant state **A** can be thought of as the metastable liquid or solution state, and the product state **B** as the solid or crystal state.

Although p_B is the theoretically ideal choice of reaction coordinate, it is never used as it is prohibitively computationally expensive to calculate. This is because it has to be estimated by sampling over all trajectories starting from that microstate, which is too expensive to do at each step of a biased simulation, and is exacerbated by the fact that many biased sampling methods require the gradient of the reaction coordinate.

In this chapter we conduct a proof of concept study to explore a method of estimating the committor for the 2D Ising model based on the current microstate, or collective variables derived from it, such that the committor might be computed on the fly in biased Monte Carlo simulations.

6.1 Calculating the Committor

Before we can construct any models to estimate the committor with, we first need some input conditions and resulting committor values as training data to interpolate between. For our system we have chosen magnetisation reversal in the 2D Ising

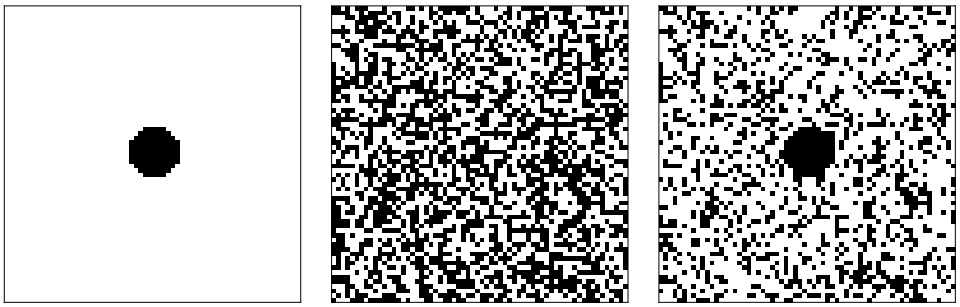
model with a grid size of 64×64 evolved with the spin flip Metropolis MC method described in 2.1.2. As this work is intended as a proof of concept, rather than a production ready tool, we choose to perform all our calculations at a single temperature $T = 1.2$ and external magnetic field $h = 0.1$, however we will discuss how this method could be extended to any range of temperatures and field strengths. In this system a microstate in which a majority of grid sites are spin down, $s = -1$, will be metastable whereas the thermodynamically stable state is for the majority of grid sites to be spin up, $s = +1$, this means there there is some free energy barrier between the mostly spin down and mostly spin up states, analogous to the nucleation of a crystal. For this system we can uniquely define each microstate by the spin values s_{ij} for each site in our 2D grid (i, j) as a grid state

$$\mathbf{s} = \begin{pmatrix} s_{00} & \cdots & s_{0L} \\ \vdots & \ddots & \vdots \\ s_{L0} & \cdots & s_{LL} \end{pmatrix}, \tag{6.1}$$

where L is the size of the matrix (here $L = 64$) and the total number of grid sites is $N = L^2 = 4096$. Here each element s_{ij} can only be $+1$ if the site is spin up or -1 if the site is spin down.

To begin, we need to generate a selection of starting conditions between the majority spin up and spin down states from which to compute the committor. We do this by producing a selection of grid states starting with a base spin up probability (of either $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7$ or 0.8) with all other sites being spin down, and inserting a circular or square central seed of various sizes. Examples of these staring grid states are shown in figure 6.1. For all of these starting grids we also compute the largest cluster size, n and number of spin up sites $N_\uparrow$ which, because we have a constant number of sites, is a proxy for magnetisation. These two collective variables give us examples of local n and global $N_\uparrow$ variables which is identified as a desirable combination in [1]. This gives us the option to estimate the committor from these collective variables as well as directly from the starting grid state.

To estimate the committor from any of these starting grid states we run many trajectories until they reach our nucleated state B $N_\uparrow > 3276$ ($> 80\%$ spin up) or return to the metastable state A $N_\uparrow < 820$ ($< 20\%$ spin up) and count the number of trajectories reaching each state n_B and n_A. From these counts we can calculate



(a) isolated seed (b) random background (c) seed with background

Figure 6.1: Selection of example starting grids. Grid (a) consists of an isolated spin up seed, radius $r = 5.5$, with a spin down background. This results in a largest cluster size of $n = 97$, and $N_\uparrow = 97$ spin up sites. Grid (b) consists of a random background, with a spin up probability of $P_\uparrow = 0.5$, resulting in a largest cluster of $n = 285$, and $N_\uparrow = 2014$ spin up sites. Grid (c) consists of a seed, of radius $r = 5.5$, in a random background with $P_\uparrow = 0.3$. This results in a largest cluster of $n = 111$, and $N_\uparrow = 1271$ spin up sites.

the committor as

$$p_B = \frac{n_B}{n_A + n_B}. \tag{6.2}$$

As each of these trajectories are independent samples of the committor, the results will form a Bernoulli distribution and so the variance of the trails will be $\sigma^2 = p_B(1 - p_B)$ giving us an error on our estimate of

$$\epsilon_{p_B} = \sqrt{\frac{p_B(1 - p_B)}{N_{\text{Samples}}}}. \tag{6.3}$$

For all of our sampled starting grids we run $N_{\text{Samples}} = 7168$ independent samples. While this may seem arbitrary, the number of samples is influenced by the compute capability of the Nvidia P100 GPU used to run our Ising model simulations. The code used to generate the results is available in reference [44] and is based on reference [45]. Due to practical limits on the time available, we draw randomly from our selection of starting grids in order to get a sampling of the possible starting microstates.

Obtaining an even distribution of committor values reduces masking effect when calculating the mean squared error (MSE) between the training data and some

prediction. This error can occur as we could have small errors in highly sampled regions (e.g. high and low p_B) and large errors in less samples regions ($p_b \approx 0.5$). This would mean the higher number of small errors would mask the larger errors in the less sampled region.

6.2 Estimation Methods

Now we have a selection of starting states and the resulting committors as training and testing data, we can construct models to predict the committor from the starting state. To achieve this we have a variety of options. Here we will discuss those we intend to use.

6.2.1 Regression

We have used regression many times in other sections to fit known, or assumed, functional forms to our data. Providing all input variables are independent we can use a linear regression model to fit this data. This model takes the form

$$y = \mathbf{X} \cdot \boldsymbol{\beta} + \epsilon \tag{6.4}$$

with y being the dependant variable, $\mathbf{X}$ being a vector of independent input variables ($\mathbf{X} = (1, x_1, x_2, \ldots, x_n)$), $\boldsymbol{\beta}$ being the vector of fitting parameters ($\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \ldots, \beta_n)$), and ϵ being some random noise.

The simplest form of this is a 1D linear regression, or straight line fit

$$y = \beta_0 + \beta_1 x + \epsilon. \tag{6.5}$$

Given a selection of samples for x and y we can estimate the fitting parameters β_0 and β_1 by adjusting them to minimise the mean squared error (MSE),

$$\text{MSE} = \frac{1}{N} \sum_{i}^{N} [y_i - (\beta_0 + \beta_1 x_i)]^2. \tag{6.6}$$

There are various methods of achieving this minimisation which we will not discuss here except to say that they can be extended from this simple 1D case to the more general case of equation 6.4.

One specific regression model we shall use is polynomial linear regression, in which we take the model to be of the form of a degree m polynomial,

$$y = \beta_0 x^0 + \beta_1 x^1 + \cdots + \beta_m x^m + \epsilon. \tag{6.7}$$

This can be extended from the single independent variable to any number of independent variables n, giving

$$y = \beta_{0,1} x_1^0 + \cdots + \beta_{m,1} x_1^m + \cdots + \beta_{n,0} x_n^0 + \cdots + \beta_{n,m} x_n^m + \epsilon. \tag{6.8}$$

For our purposes we only need to consider the case of 2 independent variables, the size of largest cluster n and number of spin ups $N_\uparrow$.

6.2.2 Artificial Neural Networks

The regression model already discussed allow us to fit any linear polynomial function of our input variables but what about non-linear functions? Instead of assuming functional forms and using standard non-linear regression models we will attempt to use an artificial neural network (ANN) to fit our data.

At their core ANNs, are essentially non-linear functions, constructed from layers of nodes. Typically, each node takes a number of inputs $\mathbf{X}$, combines them with some weights $\mathbf{w}$, and adds a bias b. This combination if then fed into an activation function f_{Activate} to give the output of the node y. This process is shown pictorially in figure 6.2. This can be represented mathematically as

$$y = f_{\text{Activate}} \left(\sum_i w_i x_i + b \right). \tag{6.9}$$

The activation function can be any single input function. Some useful examples are a simple linear function $y = x$, sigmoid function $y = (1 + e^{-x})^{-1}$, and the rectified linear function $y = \min(0, x)$. The linear activation function is often used as the final node in a regression network as it allows any output value to pass through. The sigmoid function emulates the activation of biological neurons and is bounded between 0 and 1. The rectified linear function is commonly used as systems with enough nodes and layers can theoretically be used to represent any single valued function as two layers can be used to define an input range and a third can be used to give a linear range of values within that range.

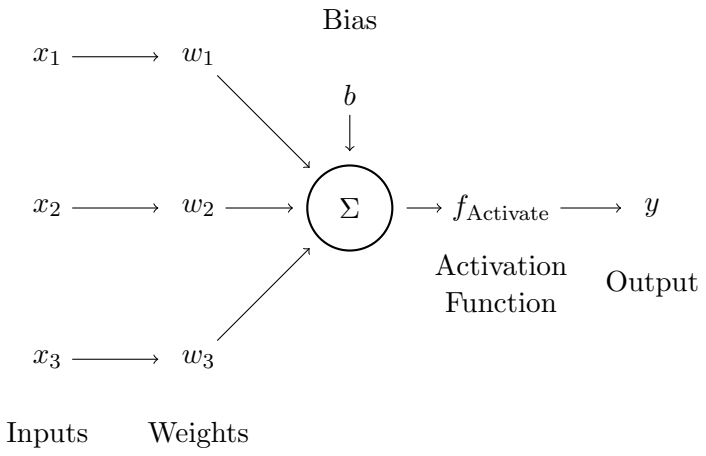


Figure 6.2: Schematic view of an individual neuron in an artificial neural network taking in an input $\mathbf{x} = (x_1, x_2, x_3)$ with some weights $\mathbf{w} = (w_1, w_2, w_3)$, applying a bias b and outputting through some activation function f_{Activate}.

These nodes are arranged into layers and, in fully connected networks, the output of every node in one layer is connected to the input of every node in the next layer, an example is shown in figure 6.3. A network can contain any number of hidden layers (layers between in the input and output layers) each of which can have any number of nodes, the only limit to this is placed on the input and output layers which represent the input and output data and so must have the appropriate dimensions.

Initially, the weights and bias for each node are set randomly and the network is trained using some known data. This training is accomplished using back-propagation and stochastic gradient descent neither of which we will discuss in detail here, but are described in [46]. This process involves calculating the outputs for a given set of inputs and computing the mean squared error, or other loss function, with respect to the expected output. This error is then propagated backwards through the network and gradient descent is used to adjust the weights and biases. The back-propagation process is repeated several times, referred to as epochs, so the network learns to approximate the output from the inputs. In practice the data is passed through the network in batches to speed up the training process.

This method of using an ANN to identify the committor is similar to work by Ma and Dinner [47]. However, they examine a different system, alanine dipeptide,

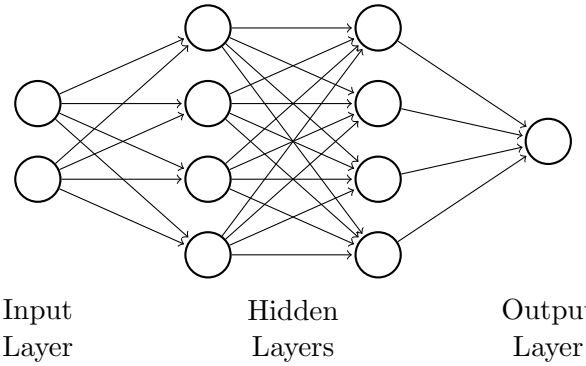


Figure 6.3: Schematic view of an artificial neural network.

and have a different methodology to the one we shall employ. They take a selection of variables and combine them in groups of three into small ANNs with the 3 inputs and a single layer of 2 neurons before the output layer. They also take a generative approach to fitting the network parameters. This generative approach starts with a collection of randomised networks for each trio of input variables. These networks are used to estimate the committor and ranked accordingly. A subset of the networks are selected, weighted by their ranking, and they are then duplicated and mutated. This process, analogous to natural selection, is repeated until the desire fitness is required. The approach is an alternative to the back-propagation method we employ. To our knowledge, no previous attempt has been made to compute a crystallisation committor for the 2D Ising model using artificial neural networks.

Over-fitting

As the number of neurons and layers increase, the number of fitting parameters rapidly increases, this can make it easy to over-fit a neural network. Doing so makes the network perform very well on the training data but very poorly on other data, in the same way as over-fitting any function. To identify if the network is over-fit, we split our data into three sets, a training set used to fit the parameters, a test set which we use to monitor the if the model is over-fit during training and tune hyper-parameters, and a validation set use to check for over-fitting after the hyper-parameters have been set. These sets should have a similar distribution of input and output parameters. At the end of each training epoch, we evaluate the loss function for the test data. If the model is behaving well then the loss for the two data sets

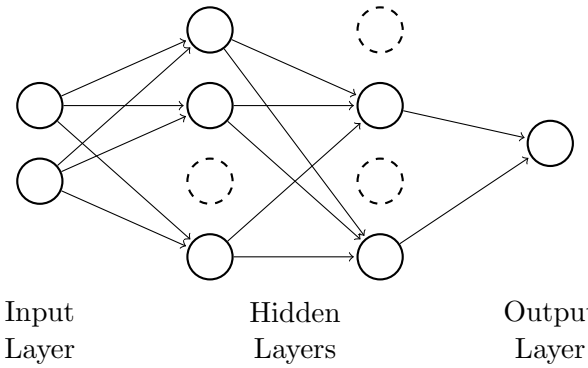


Figure 6.4: Schematic view of an artificial neural network with dropout. Each neuron is represented as active, full circle, or inactive, dashed circle.

should be similar, however if the model is over-fitted, then the loss function for the training set will be significantly lower than the test set.

The test data set allows us to identify over-fitted models but does nothing to combat it, for this we use the dropout method [48]. In this method during training we only activate neurons during each training batch with some probability. This means at each fitting stage only a subset of the neurons, and therefore fitting parameters, will be active, shown pictorially in figure 6.4. The deactivation probability is defined in term of a dropout rate. Predictably, systems with a larger dropout rate will require more training epochs to fit.

Convolutional Neural Network

So far we have only discussed ANNs for fitting a set of numerical inputs to a numerical output which is adequate for calculating the committor in terms of system collective variables, but we would like to estimate it directly from the grid state $\mathbf{s}$ of our Ising model. To do this we will use a convolutional neural network (CNN), which are widely used in image processing and computer vision. CNNs take some higher dimensional structure as an input, in our case a 2D Ising grid, and use a series of convolutional layers to filter over the structure and attempt to identify features which can then be fed into a neural network of the type already described.

These layers consist of a kernel, a feature map size and an activation function. The kernel is a matrix with the same number of dimensions as the input, but usually a different size which is convolved with the input and each element of this convolution

passed through the activation function. For example in our 2D Ising model we have a grid size of 64×64 but will use a 3×3 kernel for our convolutional layers. The feature map size is the number of independent kernels in the convolutional layer. The elements of the kernel for each feature is determined by the back-propagation process as the for neurons discussed previously. After the output of the last convolutional layer in a model, the output grids of the features maps are flattened to a 1D array which is then used as the input for layers of neurons like those discussed in the previously.

Estimating Errors in Neural Networks

Up to now we have neglected to discuss how we can determine an error bar or confidence interval for results from our ANN models. As well as giving an indication of how confident we should be in any prediction, this also provides a practical limit on training as there is little reason to train a model to be more confident in its predictions than we are in the training data. Some machine learning models, including Gaussian process regression and Bayesian neural networks, are constructed from probability distributions for each parameter and so can readily determine their confidence in a given result. This is not the case for ANN and CNN which can confidently predict a wildly inaccurate result for unseen data if not used carefully. Various approaches exist to counteract this inbuilt over-confidence. Here we will discuss three of them, bootstrapping [49], mean variance estimation [50], and MC dropout [51].

In this context bootstrapping [49] works in a similar way to the usual statistical method, in which the result of any test or model is repeated multiple times with data re-sampled (with replacement) from the original data set. This allows a mean and standard error to be computed for any process for which a distribution of input data exists. To apply this to ANNs, we simply train several networks using different subsets of the training data. This method has the obvious downside that it increases the training time by a factor or the number of subsamples required.

Mean variance estimation [50] uses two separate ANNs. The first of which is trained normally on half of the data set to give a mean value prediction, this network is then used to predict mean values for the second half of the data set. These predictions are used to calculate the variance between the predictions and the training data, and the second network is trained to approximate that variance.

This method requires roughly double the training time for the ANN, which is significantly less than bootstrapping, and also introduces the possibility of over-fitting, or overconfidently reporting, the variance as well as the mean.

The third method we will discuss in MC dropout [51]. As its name suggests, this method uses the dropout method discussed earlier. In this method, we train our ANN as usual using the dropout method. The dropout rate would normally be set to zero when using the network for predictions so that all of the neurons are active giving the best estimate of the mean. However for MC dropout we leave the dropout rate at a non zero value and sample the result for one data point several times. Because of the stochastic nature of the dropout method we will generate a distribution of results from a single input. This distribution allows us to calculate a mean and standard error for our prediction. This method doesn't require any extra training time, however it does require multiple evaluation of the model for each input.

All of these methods have some trade off in terms of training and evaluation time and the appropriate choice will depend on the application as well as the priorities on the implementation, for example MC dropout would probably be a poor choice for a system that has to evaluate inputs quickly as the model needs to be evaluated many times per input, whereas the increased training time of bootstrapping may be a bad choice for large models, as the train can become computationally expensive.

6.3 Fitting For The Committor

Before we do any fitting, we should examine our data set, the distribution of parameters for which is shown in figures 6.5 and 6.6. Figure 6.5 suggests some structure in our phase space coverage. This is due to the method we used to select our stating microstates, as we have fixed clusters surrounded by a bulk with one of several spin up probabilities. These discrete probabilities lead to the multi-peaked $N_\uparrow$ distribution, and the clusters in a entirely spin down bulk give rise to the $n = N_\uparrow$ points. From figure 6.6 we can see that we have an abundance of samples with committors of $p_B \approx 0$ and $p_B \approx 1$. As we discussed earlier we ideally want an even distribution in our output variable samples, this is because in an uneven distribution means that a poor fit in a poorly sampled region can be masked by a good fit and a well sampled region when considering the MSE of the data set. To achieve a more even

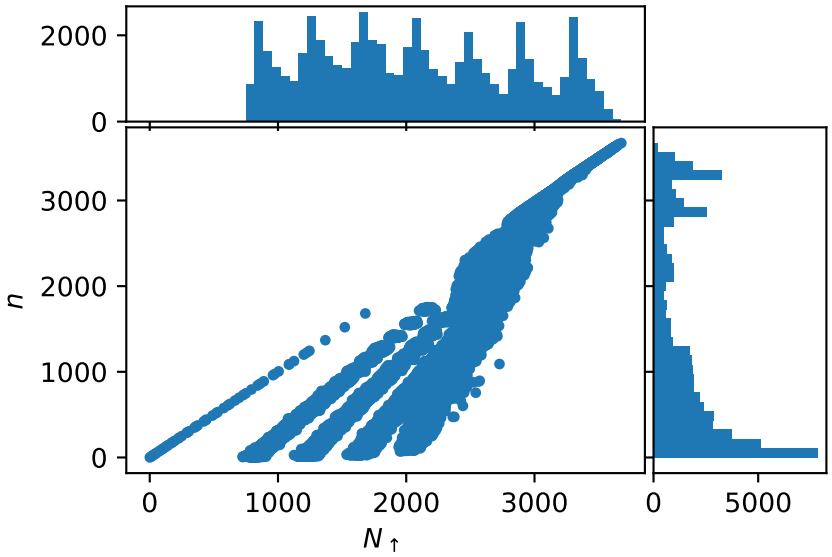


Figure 6.5: Phase space coverage of complete committor data set.

distribution, we separate the samples where p_B is above or below some thresholds, $p_{\text{low}} = 0.05$ and $p_{\text{high}} = 0.95$, we then randomly draw a smaller number of samples from these regions. We choose to take $N_{\text{low}} = 100$ samples from the lower region and $N_{\text{high}} = 1000$ samples from each region, the larger number of samples from the higher region is because that region represents a larger range of input states. This sub sampling method may seem a bit arbitrary, but we would like to use as much of our generated data as possible and as our stating microstate selection is physically motivated, seed nuclei in some background spin up density, it seems reasonable to allow some of this sampling bias within some limit.

6.3.1 Linear regression

As our first estimation method, we will use a polynomial regression to fit the committor as a function of out two collective variables, n and $N_\uparrow$. For this we use a polynomial degree of 6, the predictions of this model are shown in figures 6.7, 6.8 and 6.9, with the MSE for these predictions being 0.0143 ± 0.0003. Some of the predictions of this model fall outside of the range $p_B \in [0, 1]$. The model performs poorly for the microstates with $N_\uparrow < 500$, these microstates consist of a single spin up seed, surrounded by an all spin down background. There are much fewer of these

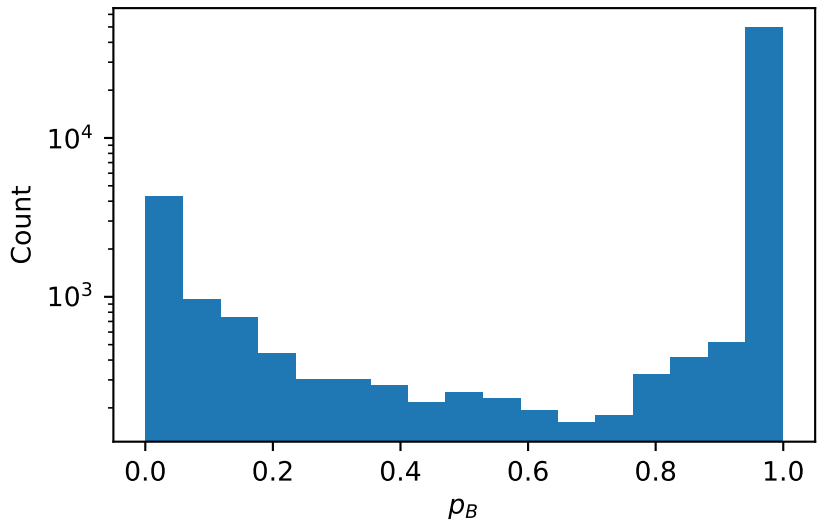


Figure 6.6: Distribution of p_B in data set.

microstates in the data set, as an all spin down background can only be represented one way, whereas for the microstates with an background with a spin up in the interval $p_\uparrow \in (0, 1)$ these are many ways to fulfil that criteria. This means the strictly isolated seeds will only have one sample in our data sets where as we may have up to 1000 samples for any other background probability. Figures 6.8 and 6.9 also show the multi-valued nature of p_B with respect to either n or $N_\uparrow$, that is that a single value of n (or $N_\uparrow$) can represent multiple values of p_B. This means that p_B cannot be fully described by either n or $N_\uparrow$ alone.

6.3.2 ANN - Collective Variables

For our first ANN model we will take our two collective variables as inputs to our network. To avoid the weights and biases getting too large, it is advisable to scale inputs between 0 and 1. For our inputs a simple, physically motivated scaling is to represent them as a fraction of the total number of grid sites. For each of our neurons, other than the input and output, we will use rectified linear activation functions.

When using an ANN for regression, it is common to use a linear activation function for the output neuron as it places no limits on the output of the network.

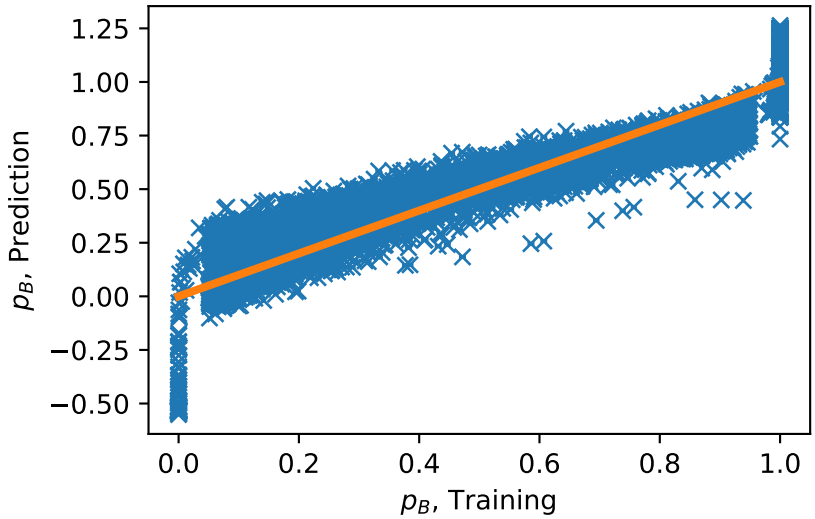


Figure 6.7: Comparison of training data and predictions of p_B for a polynomial linear regression degree 6 based on large cluster size n and number of spin ups $N_\uparrow$. Orange line indicates "correct" prediction, prediction = training.

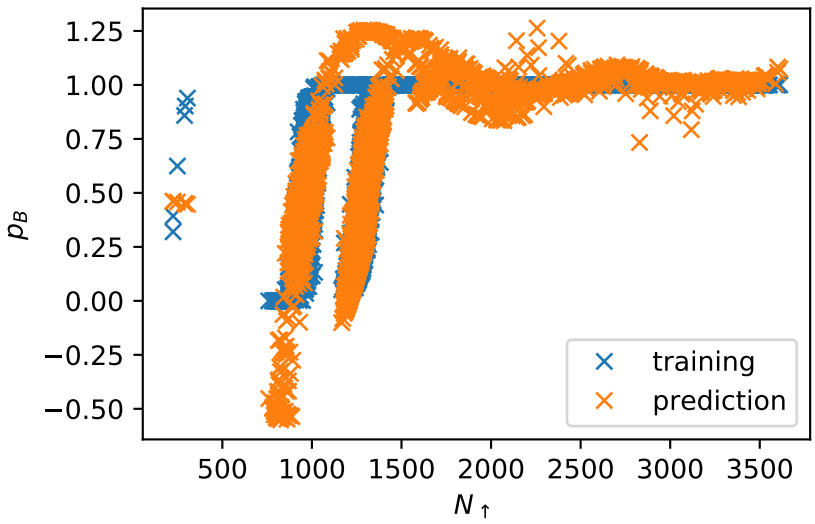


Figure 6.8: Comparison of training and predicted values of p_B for polynomial linear regression degree 6 based on largest cluster n and number of spin ups $N_\uparrow$. Comparison shown against a single input variable, $N_\uparrow$.

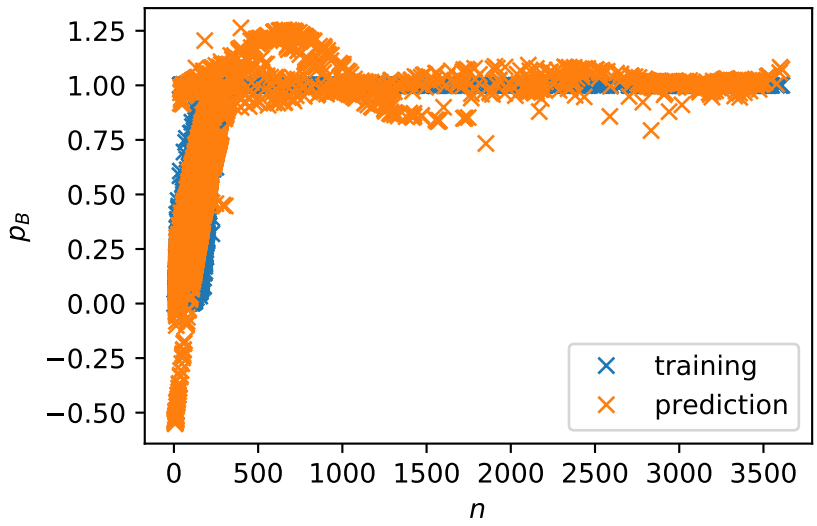


Figure 6.9: Comparison of training and predicted values of p_B for polynomial linear regression degree 6 based on largest cluster n and number of spin ups $N_\uparrow$. Comparison shown against a single input variable, n.

In this case, however, our output parameter is bounded between 0 and 1, this makes the sigmoid function a good choice for our output as it is also bounded between 0 and 1. For our loss function we will use the mean squared error, and to compute errors in our predictions we will use the MC dropout method.

We have to set some hyper-parameters for our models, those are the number of layers, the number of nodes per layer and the dropout rate. To determine suitable values for these parameters, we first must divide up our data set randomly into three sub sets for training, testing and validation containing 56.25%, 18.75% and 25% of the complete data set respectively. (This split is the same for all of our ANN based models). The training set is the data we will use to fit the models, the testing data set will be used as an initial check to over-fitting and for selecting hyper-parameters, and the validation data set will be used to check for over-fitting in the final model.

As this work is mostly a proof of concept, we will not conduct a detailed analysis for our number of layers and nodes per layer. For this we simply fitted the model for a selection of layer numbers and sizes and used the one for which the mean squared error of the test data set was the smallest. The parameters selected were 4 layers of 8 nodes followed by the single output node.

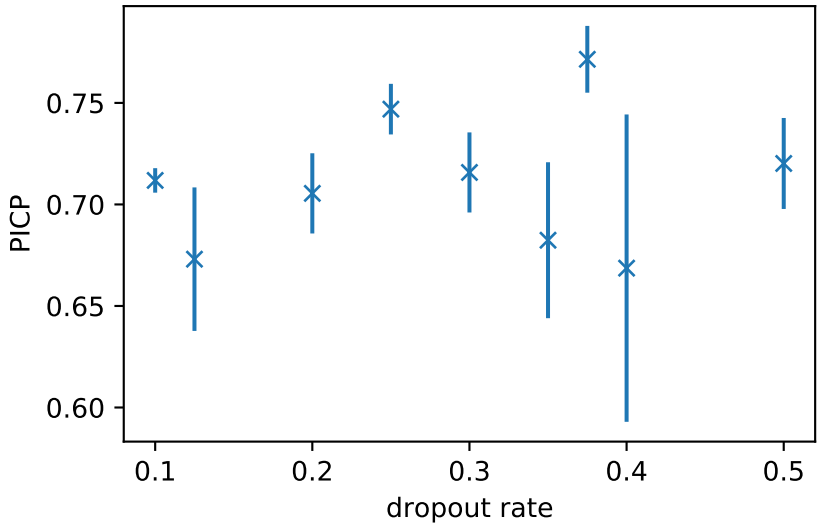


Figure 6.10: Prediction interval coverage probability of the artificial neural network model for the testing data set.

To choose the value of the dropout rate, we conducted a slightly more involved, but still only cursory analysis. For this, we trained the model using a variety of dropout rates. These models were then used to predict the committors for the test data set using 10 samples per input. We then calculated the mean prediction as well as the standard deviation of prediction for each input, the range of 1 standard deviation will be our prediction interval. We can then calculated the prediction interval coverage probability (PICP), which is the probability that any prediction will fall within the prediction interval, and the mean prediction interval width (MPIW) which for our purposes is the average standard deviation of predictions. The PICP and MPIW are shown in figures 6.10 and 6.11. We want to identify a dropout rate which maximises our PICP while minimising out MPIW, that is we want as many of our predictions as possible to fall within the smallest range possible. For our dropout rate we choose $p_{\text{dropout}} = 0.2$.

Now we have decided on our hyper parameters, we can train our model and examine the results. We conducted some manual testing and determined that a training time of 100 epochs gave a reasonable balance of run time, MSE for the training and test data sets, and fluctuations between epochs of the MSE. Figure 6.12

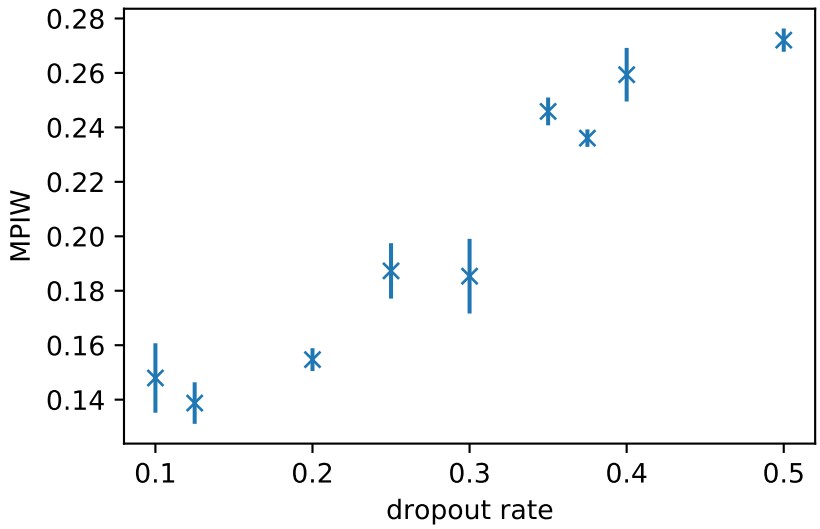


Figure 6.11: Mean prediction interval width of the artificial neural network model for the testing data set.

shows the mean squared error for each of the data sets. This shown that the MSE is similar for all of the data sets which indicates that our model is not over-fitted.

For our training data set, figure 6.13 shows a comparison of the committor value used for training and the result predicted by the final model. Figures 6.14 and 6.15 show the deviation of the predicted mean from the training value for $N_\uparrow$ and n respectively.

These figures show that our model is over-estimating low values of p_B in general. We also have a selection of outliers with very low values of $N_\uparrow$. These starting grids consist of a spin up seed with the rest of the grid being spin down. These grid states may be fitted poorly because we have very few samples of them relative to the other starting conditions, as discussed previously. There are also a group of outliers at $N_\uparrow \approx 1700$ for which the model under-predicts p_B. These grids all appear to have a low value of n, some of which are just background and some have small seeds.

Figures 6.16, 6.17 and 6.18 show the same plots for the validation data set. These plots show outliers in similar locations to the training data, and look broadly similar again indicating that the model has not been over-fitted.

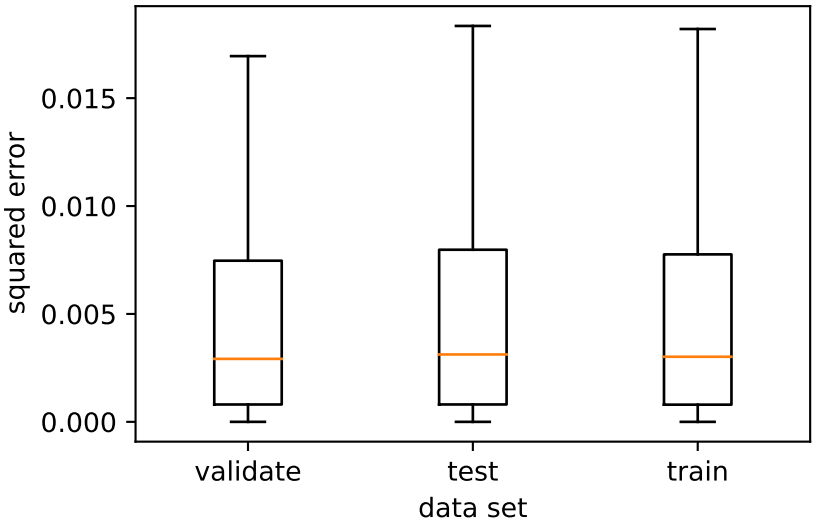


Figure 6.12: Squared error distribution for the ANN collective variable model predictions for all 3 data sets. For these box plots any values 1.5 times the inter quartile range from the mean are classified as outliers and not shown.

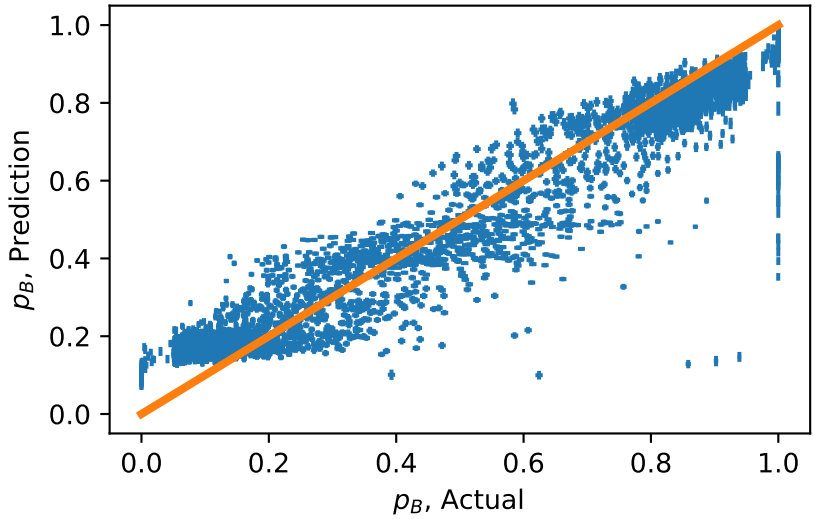


Figure 6.13: Comparison of the values of p_B predicted by the ANN collective variable model to the actual data set values for the training data set.

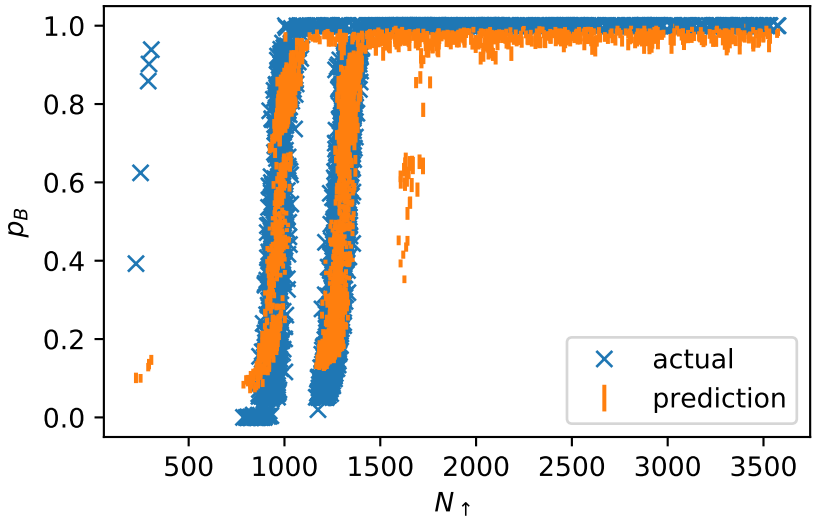


Figure 6.14: Comparison of predicted and actual data set values of p_B for the ANN collective variable model for the training data set focusing on the value relative to one of the two variables, number of spin up sites $N_\uparrow$.

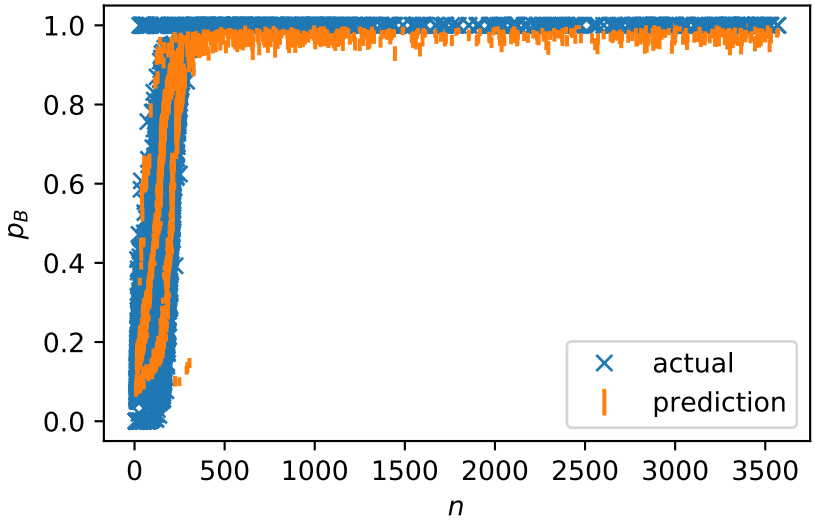


Figure 6.15: Comparison of predicted and actual data set values of p_B for the ANN collective variable model for the training data set focusing on the value relative to one of the two variables, size of largest cluster n.

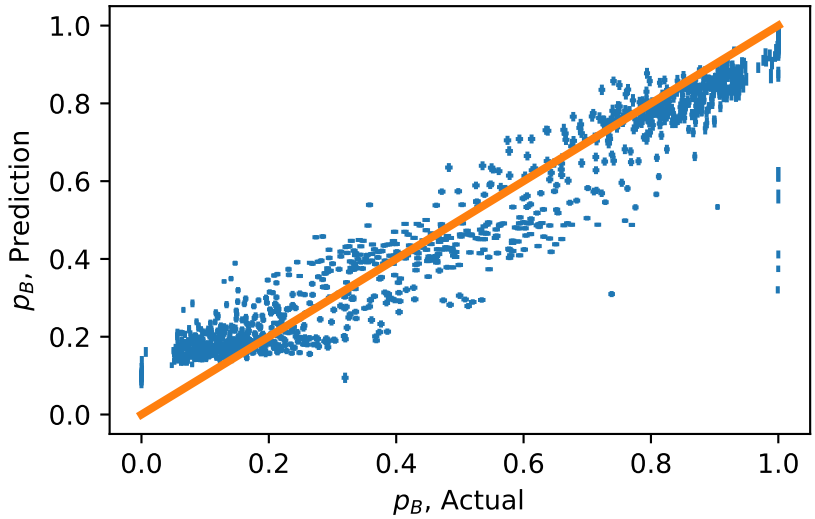


Figure 6.16: Comparison of the values of p_B predicted by the ANN collective variable model to the actual data set values for the validation data set.

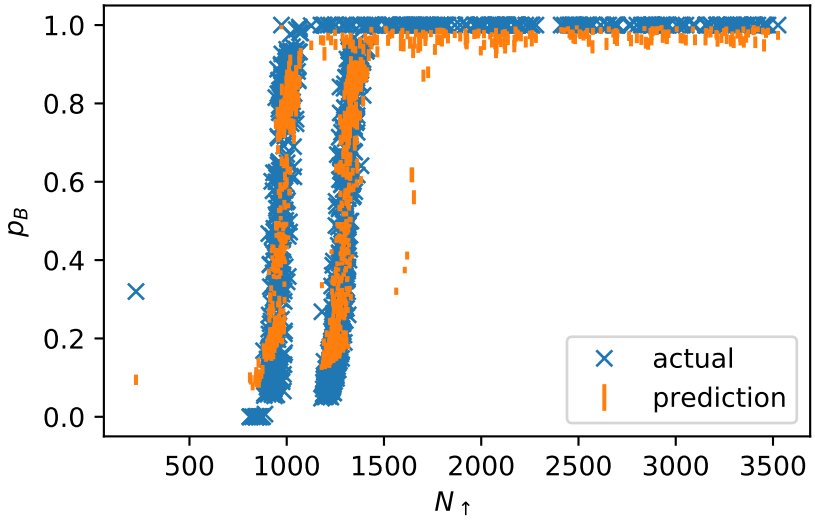


Figure 6.17: Comparison of predicted and actual data set values of p_B for the ANN collective variable model for the validation data set focusing on the value relative to one of the two variables, number of spin up sites $N_\uparrow$.

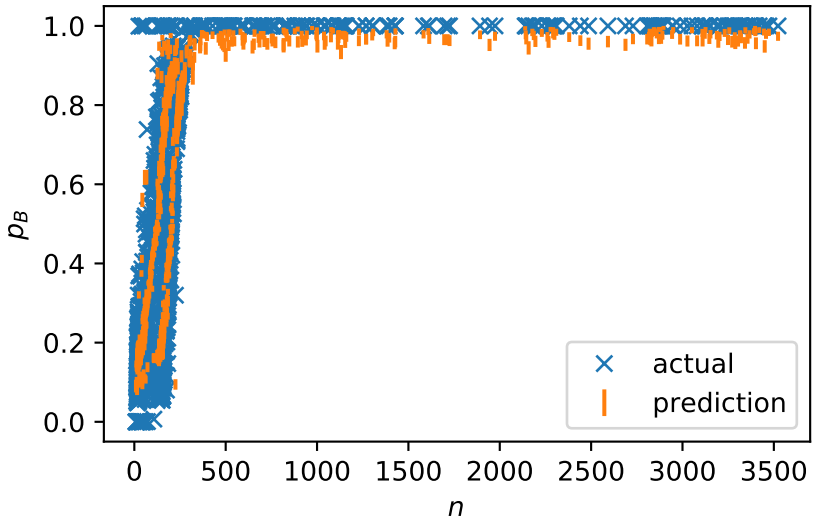


Figure 6.18: Comparison of predicted and actual data set values of p_B for the ANN collective variable model for the validation data set focusing on the value relative to one of the two variables, size of largest cluster n.

6.3.3 CNN - Grid State

In the last section we used collective variables as the input for the network, but all of the information about the system is contained in the grid state $\mathbf{s}$, so we should be able to use this grid state as the input for our ANN. To do this we will slightly alter our representation of the grid state to 1 for spin ups and 0 for spin downs, rather than 1 and -1, as this is the method we use for storing the grid state in our database and so removes the need for a conversion step before inputting to the network. To allow our ANN to accept this input, we will use convolutional layers, discussed earlier in section 6.2.2. These layers reduce our 2D grid into a 1D array of parameters which will be passed to layers similar to the ones used in the previous section.

The setup for this model is as follows,

1. three 3×3 convolutional layers with feature maps of size 16, 32 and, 64,
2. a layer of 16 neurons, the output of which is normalised for the batch,
3. three layers of eight neurons,
4. a single output neuron with a sigmoid activation function.

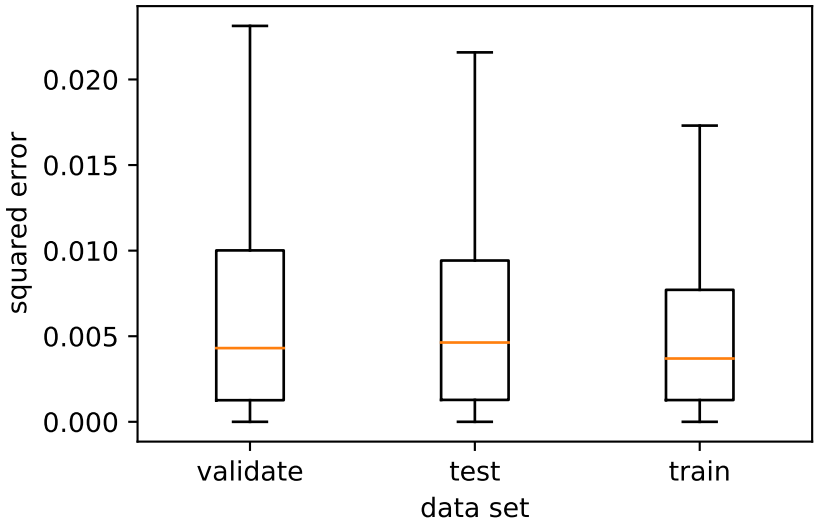


Figure 6.19: Squared error distribution for the CNN grid state model predictions for all 3 data sets. For these box plots any values 1.5 times the inter quartile range from the mean are classified as outliers and not shown.

The activation function for all neurons other than the output is the rectified linear function. The dropout rate used for training is 0.2 (20% of neurons will be inactive in each training epoch) and the MC dropout method is used to evaluated confidence intervals for the predictions, with 100 samples per predictions). This model is trained for 100 epochs with a batch size of 8. Again we are not concerned with the optimisation of the model so these parameters are based on the previous model, the grid size and some manual testing. The training time of 100 epochs as in the previous section, is a balance between getting a low MSE for the training and test data sets, reaching a point where the MSE is similar between epochs, and run time.

Figure 6.19 shows the mean squared error for the training, testing and validation datasets. Here we see that the training set has a slightly lower MSE than the testing and validation data sets. This indicates some degree of over-fitting, however, the MSE in this figure is based on the predictions after our MC dropout sampling, rather than the individual predictions which is the MSE used for updating the model. The sampling has the effect of exaggerating difference in the MSE between the data sets.

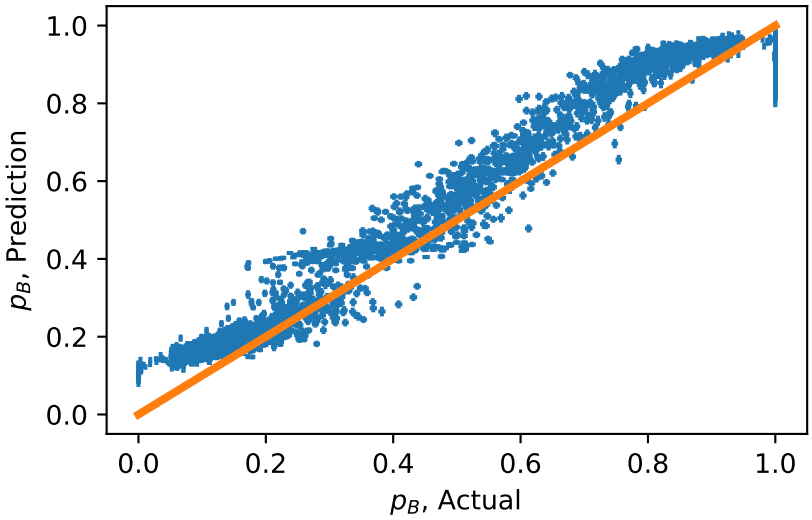


Figure 6.20: Comparison of the values of p_B predicted by the CNN grid state model to the actual data set values for the training data set.

The predictions for this model are shown in figures 6.20, 6.21 and 6.22 against the training value of p_B, $N_\uparrow$ and n respectively. These figures show an over-estimation of p_B for low values, and a lower estimation at higher $N_\uparrow$. This training set only contains one of the grids with a seed surrounded by spin downs, however it has achieved a much closer prediction of p_B for this grid that the collective variable model.

Figures 6.23, 6.24 and 6.25 shown the same plots for the validation data set. This data set contains more of the isolated seeds and again the model does a good job of predicting p_B for them. We again see that low values of p_B are over-estimated and there are large fluctuations in the predicted p_B at high $N_\uparrow$. The deviations at $N_\uparrow \approx 1700$ are present in the data set but not the training set.

6.3.4 Mixed Neural Network - Grid State and Collective Variables

The two previous models used the gird state and collective variables separately, now we will combine them into one mixed model. To do this we process our grid through the convolutional layers then one neuron layer, and we process the collective variables through 2 layers, before combining then and processing them through an additional 3 layers.

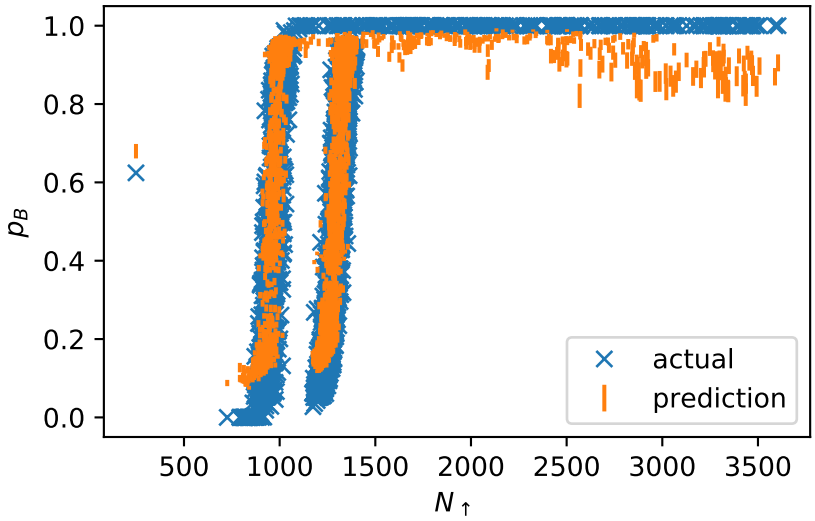


Figure 6.21: Comparison of predicted and actual data set values of p_B for the CNN grid state model for the training data set. Comparison is with respect to the number of spin ups in the starting grid state $N_\uparrow$.

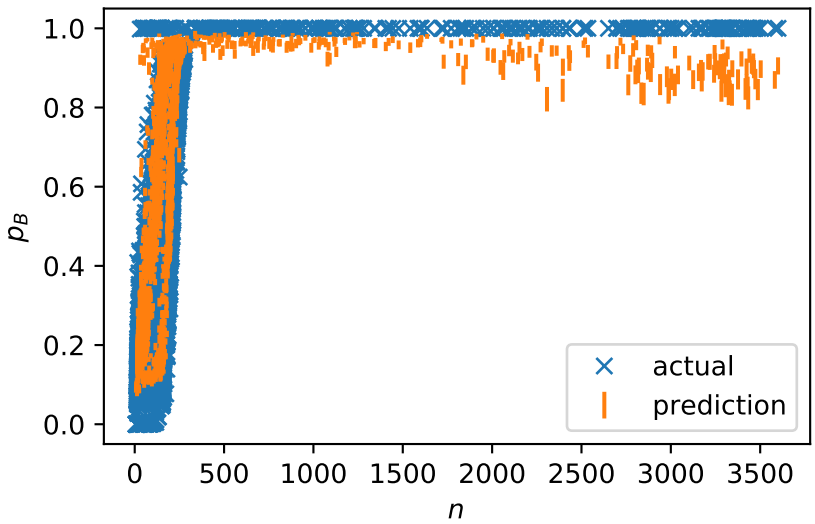


Figure 6.22: Comparison of predicted and actual data set values of p_B for the CNN grid state model for the training data set. Comparison is with respect to the size of largest cluster in the starting grid state n.

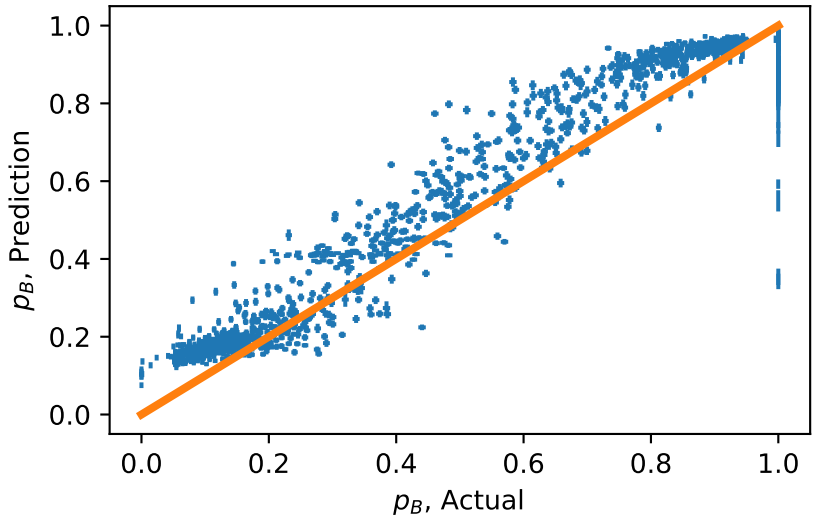


Figure 6.23: Comparison of the values of p_B predicted by the CNN grid state model to the actual data set values for the validation data set.

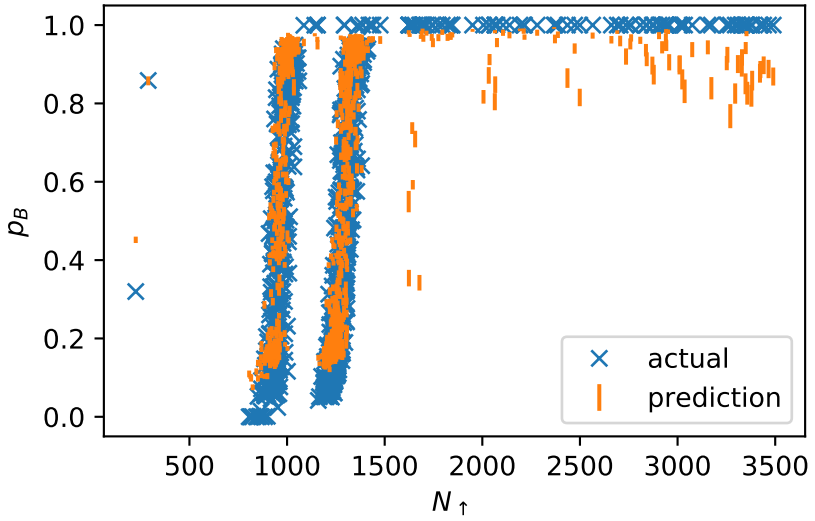


Figure 6.24: Comparison of predicted and actual data set values of p_B for the CNN grid state model for the validation data set. Comparison is with respect to the number of spin ups in the starting grid state $N_\uparrow$.

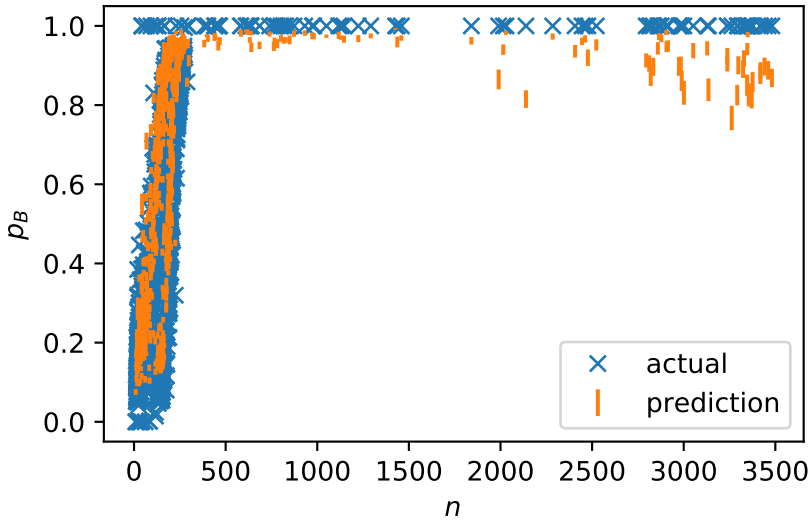


Figure 6.25: Comparison of predicted and actual data set values of p_B for the CNN grid state model for the validation data set. Comparison is with respect to the size of largest cluster in the starting grid state n.

For the grid processing we use the same combination of layers as before, stopping after the 16 neuron layer. For the collective variable processing we process our 2 inputs through 2 layers of 8 neurons. We then combine the two parts and process then with 3 layers of 8 neurons before outputting through a single neuron. As before all of the activation function are rectified linear functions, except for the output layer which is a sigmoid function. This model is trained for 100 epochs with a dropout rate of 0.2 and the confidence interval is determined using MC dropout. The MSE for the training, test and validation data sets is shown in figure 6.26. As with the grid state model, the MSE indicates that the model is over-fitted with the MSE for the training data set being smaller than the test and validation data sets. Again this MSE difference is exaggerated by the MC dropout sampling.

The predictions of this model for the training data set are shown in figures 6.20, 6.21 and 6.22 against the training value of p_B, $N_\uparrow$ and n respectively. With the same plots for the validation data set shown in figures 6.20, 6.21 and 6.22. From these figures we see that, as with the other models, our predictions are worst for the isolated seeds and we have some deviations at $N_\uparrow \approx 1700$.

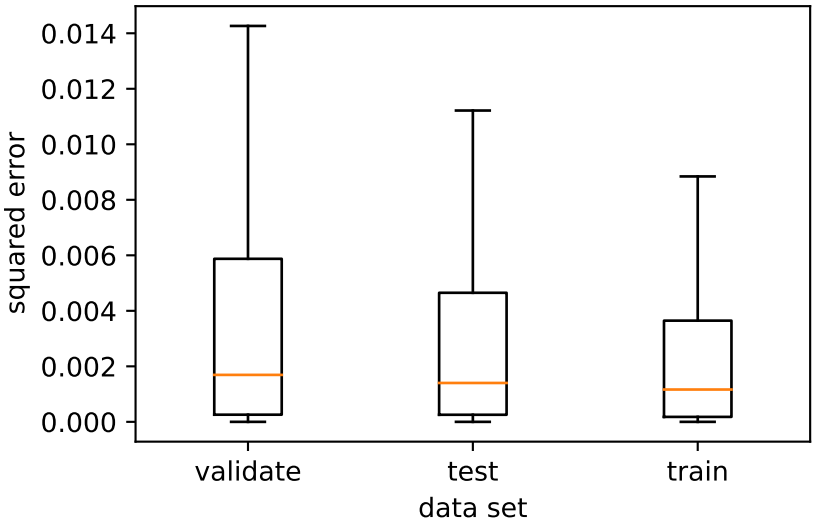


Figure 6.26: Squared error distribution for the mixed grid state and collective variable model predictions for all 3 data sets. For these box plots any values 1.5 times the inter quartile range from the mean are classified as outliers and not shown.

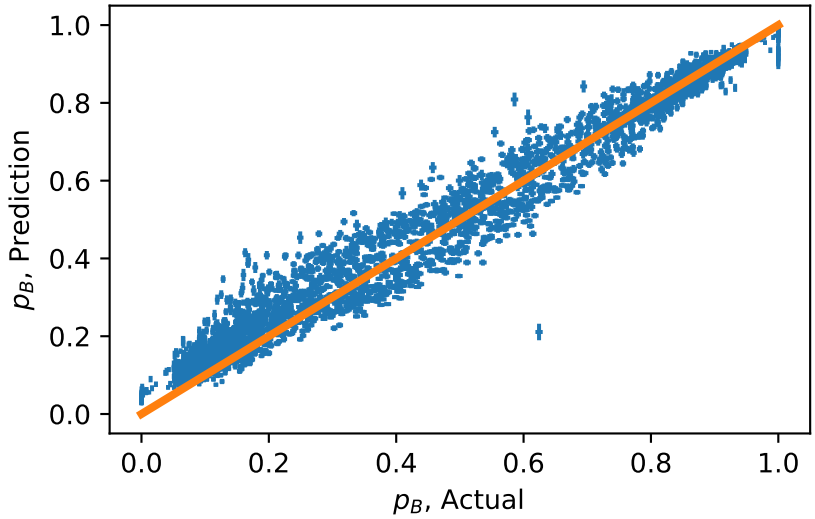


Figure 6.27: Comparison of the values of p_B predicted by the mixed grid state and collective variable model to the actual data set values for the training data set.

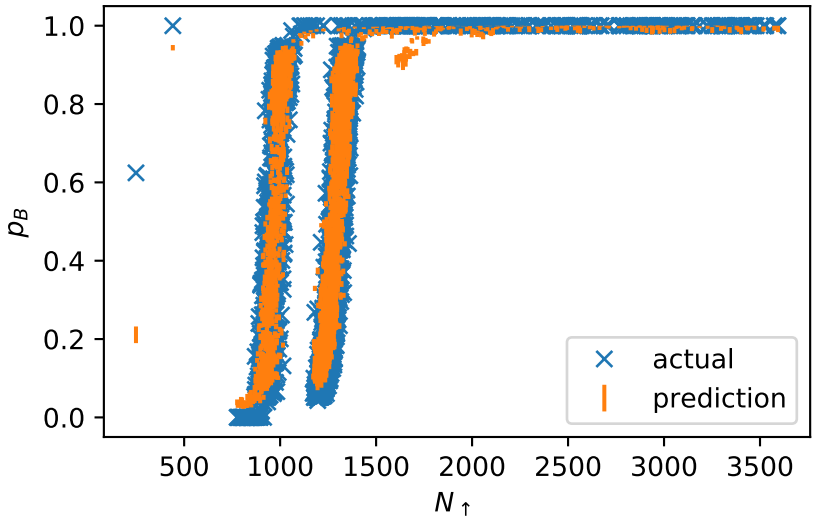


Figure 6.28: Comparison of predicted and actual data set values of p_B for the mixed grid state and collective variable model for the training data set focusing on the value relative to one of the two variables, number of spin ups $N_\uparrow$.

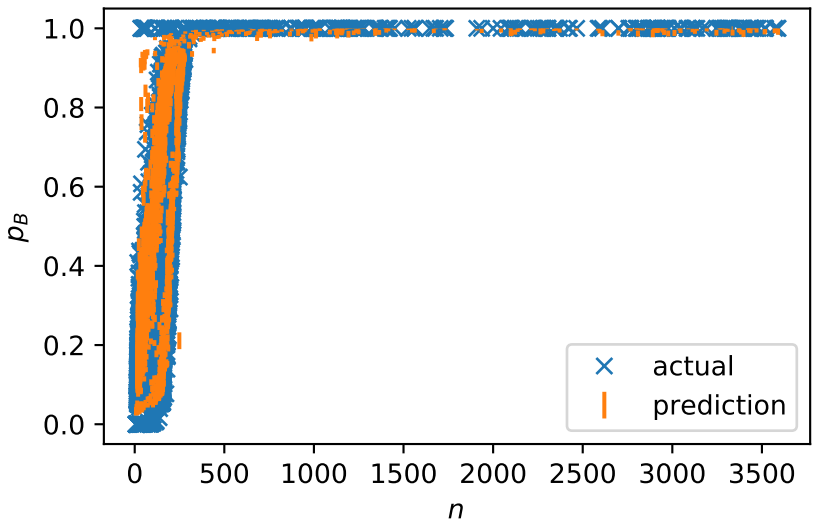


Figure 6.29: Comparison of predicted and actual data set values of p_B for the mixed grid state and collective variable model for the training data set focusing on the value relative to one of the two variables, size of largest cluster n.

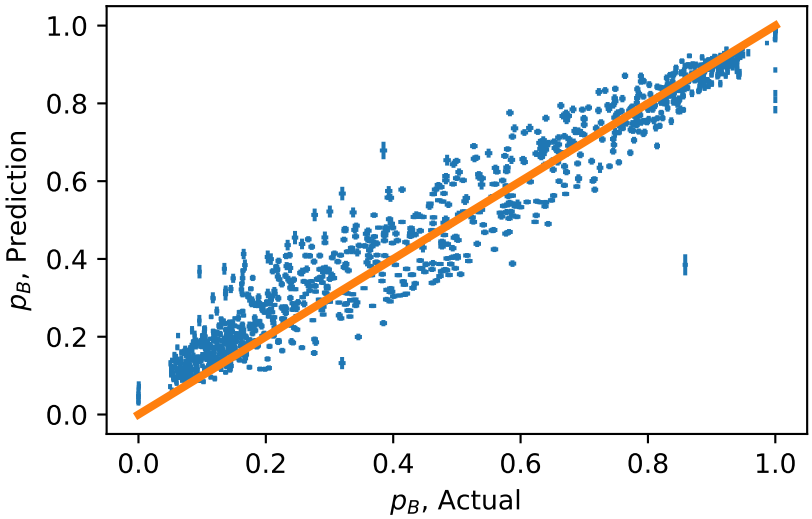


Figure 6.30: Comparison of the values of p_B predicted by the mixed grid state and collective variable model to the actual data set values for the validation data set.

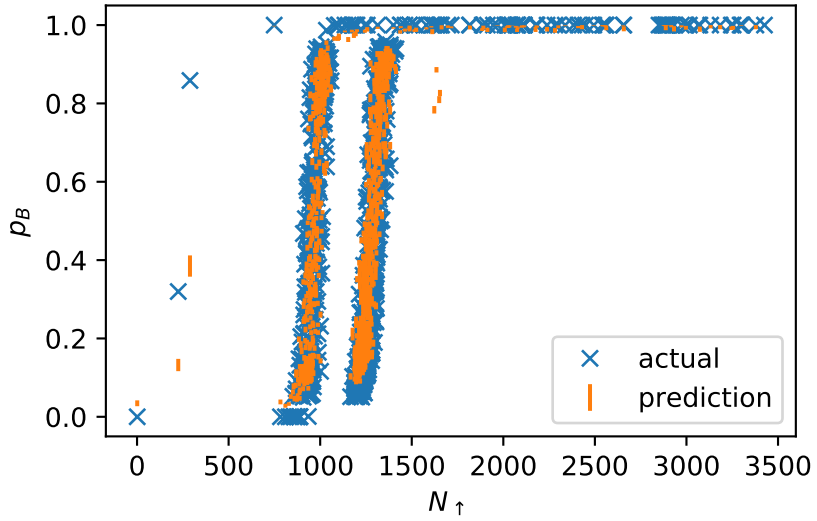


Figure 6.31: Comparison of predicted and actual data set values of p_B for the mixed grid state and collective variable model for the validation data set focusing on the value relative to one of the two variables, number of spin ups $N_\uparrow$.

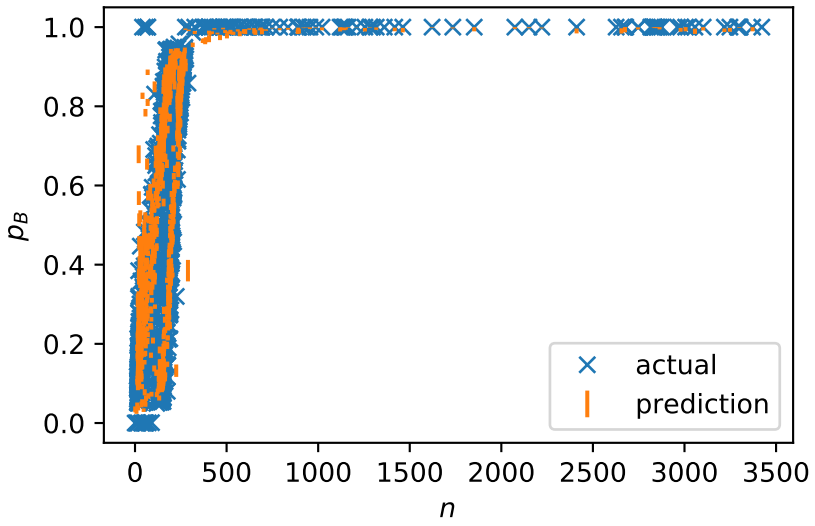


Figure 6.32: Comparison of predicted and actual data set values of p_B for the mixed grid state and collective variable model for the validation data set focusing on the value relative to one of the two variables, size of largest cluster n.

6.4 Conclusion

All of the ANN models have a smaller MSE than the polynomial regression and, while better conceived linear or non-linear models could be used in the regression, the fact that we do not need to pre-decide the exact functional form of our model is a strength of the ANN methods. The CNN model taking only the grid state as input is the best example of this, as in this model we only provide the model the grid state and the back-propagation process determines how to best determine p_B from it. Although the lack of need to predefine a functional form for the model can be a major advantage for ANN based models, if we are only interested in determining the committor, because these models are highly connected and non linear it can be difficult to understand which parameters, or collections of parameters are most influential in the output. The opacity of these models limits their usefulness if we wish to understand each parameter's impact.

Of the ANN based models, the model combining the grid state and collective variables had the lowest MSE with the narrowest range. The collective variable model had a similar MSE with a wider range, and the model based purely on grid states had a slightly larger MSE. Both of the models using the grid state were slightly over-fitted and the training data set has a lower MSE than the validation and test set. Despite this, all of the models performed reasonably well, especially considering the naivety with which their hyper-parameters where chosen. With each of these models there was an array of hyper-parameters to tune, and in this work, we only made a reasonably small effort to do so. By tuning the number of layers, neurons per layer, number of convolutional layers, and number of feature maps per convolutional layer the accuracy of predictions could be enhanced.

All of the neural network based models manage to predict committor values $0 < p_B < 1$ with some degree of accuracy. Whether these predictions are accurate enough, will depend on what they are to be used for, for example if we just want to check if a simulation has reached either state these results may be sufficient. Even with the current level of uncertainty these models may perform as a better reaction coordinate than the largest cluster size, although more work would be required to test that.

All of the data in our data set is for a single temperature and external field, however given enough data these models could also be trained to take those values as inputs broadening their use cases. These methods could also be easily extended

to the 3D Ising model and could possibly be extended to off lattice MD system, although the "grid state" models would require substantial adjustment.

Despite the success of these models, their use, and the extension of them to more dimensions, or variable temperature and external field require the committor to be calculated for a large number of samples. While we made very little effort to optimise our data set, it did take a non-trivial amount of compute time (96 hours on a single Nvidia P100) to generate the data set. It is possible that using better sampling methods to cover the required phase space could reduce that. It is also the case that the more general the model, or complex the system, the more more training data will be require. This initial compute time may be a reasonable trade off in some situations (maybe it would be preferable to train a model for the committor before running some path sampling algorithm) but there will be situations where this approach is unnecessary.

Another area of further study could be using these ANN based models as the reaction coordinate in path sampling calculations. We could then compare these results to calculations using other reaction coordinate, potentially including explicit committor calculations for a simple system. There is also plenty of scope for optimising these models, or even just the hyper-parameters. There is also the possibility of conducting analysis on the fitted weights and biases to potentially gain insight into what has the largest effects on the model's predictions.

Chapter 7

Conclusion

In chapter 4 we implemented two variations of the seeding method, which we referred to as single seed and single temperature methods, finding a reasonable agreement between the nucleation rates but with differing chemical potentials and interfacial free energies. We then examined the heat transport within the Lennard-Jones simulations by examining the temperature within shells around the cluster seed when no thermostat is applied to the simulations. This shows some heat transport in the system, taking heat released at the cluster outwards into the bulk. We then computed the thermal diffusivity for this Lennard-Jones system and modelled the thermal diffusion radially, determining that it takes $1-2$ time units for a significant temperature change to be detected at a distance of $r = 8$, which would be well within the bulk of our Lennard-Jones simulation. This diffusion of heat is closer to the situation encountered in physical experiments in which temperature is likely to be controlled from the system boundary, or at some fixed heat source.

As there is some heat diffusion through these simulations, we plotted the velocity distributions of the particles in the simulation for each of the sampled cluster sizes. From these distributions, we calculated the kinetic temperature for each cluster size to determine if the system is is a quasi-equilibrium at each n, as required by CNT. We find that the kinetic temperature is in best agreement with the thermostat temperature for the cluster sizes most frequently sampled, and we show the largest deviations are for cluster sizes which are only sampled by growing or shrinking trajectories rather than having samples going in both directions.

Overall in this chapter we find that the effects of heat transport, while ignored in most simulations, have little effect on the timescale of our seeding trajectories for

systems of this size. This means the neglecting of heat transport in these methods should not diverge the results from physical experiments, providing the timescales are short enough, and the chosen system is well described by CNT. We also find that it is important to ensure that calculations include samples from both growing and shrinking trajectories, especially in the case of the single temperature method where these trajectories are used to calculate the gradient of the free energy surface. These results suggest that, for the Lennard-Jones system we have studied, the size of largest cluster is a slow degree of freedom, as required for CNT.

In chapter 5, we use an average likelihood approach to quantify how well simulation trajectories are described by both Markovian and non-Markovian models. Here we use two memory models, one with a linear memory factor and one with a reciprocal length factor for our non-Markovian models, and validate them using 1D random walks with defined memories to determine their effectiveness. Both memory models succeed at determining the memory length of the trajectories. We then applied these memory models to trajectories generated from 2D Ising model and Lennard-Jones MD simulations like those from chapter 4. In both cases, we found that if we only allow for step sizes of 1, then they are best described by a low memory or memoryless process if we have a linear memory function or that the descriptions are indistinguishable form memoryless, if we have a reciprocal length memory function. If we allow for step sizes greater than 1, then the simulations are best described by either a short non-zero memory, if we have a linear memory function, or just a non-zero memory if we have a reciprocal length memory function. Comparing all of these results we find that the most likely description of these simulations would be a non-zero memory length while allowing for steps of size 1 or greater. If we have a linear memory function, then we would require this memory to be short (< 4), however if we have a reciprocal length memory function non-zero memories are indistinguishable between lengths of 1 and 10.

The result that the MD trajectories are best described by non-Markovian walks on the CNT free energy landscape is unexpected, especially given the results of chapter 4. The memory present in these trajectories could be the result of time dependent effects, such as the thermal diffusion discussed in chapter 4, or an indication the free energy barrier is not well described purely in terms of our chosen reaction coordinate.

In chapter 6 we use polynomial regression and neural networks to predict the committor for the 2D Ising model with spin flip dynamics under magnetisation

reversal. We attempt to fit for the committor using collective variables using polynomial regression and an artificial neural network, the current grid state using a convolutional neural network, and a combinations of the two using a neural network with both convolutional and normal layers. Here we have some success with predominate outliers be for condition where our data set had very few samples. We find that the best performing approach was the combination of the grid state and collective variables, although all of the neural network models performed well. We also discuss the extension of these machine learning models to more than one temperature, suggesting that the temperature could be provided in the same way as the mixed neural network. We also discuss the limitations of these models, noting that they do not provide insight into important parameters without further analysis and that they have a large upfront computational cost.

These models give us the ideal reaction coordinate, unlike the size of largest cluster used in chapters 4, and 5. This methodology can potentially be used to calculate the committor in systems for which we do not have good reaction coordinates, which would also allow path sampling methods, or other similar methods, to allow for comparisons to experiments. This does require that we have a method of determining if a system is in the reactant or product state, and that we can feasibly calculate the committor from enough starting states.

In this thesis we examined some of the assumptions required for CNT based seeding methods in chapters 4 and 5. In chapter 4 we satisfied ourselves that the "size of largest cluster" metric was a slow degree of freedom, and that the Lennard-Jones particles thermalised on a timescale shorter than that of the changes in cluster size. We then continued our exploration in chapter 5 testing the assumption that cluster sizes are well described by a Markovian random walk on the one dimensional free energy landscape. By considering the likelihood that trajectories were generated by a random walk with a given memory, we found that seeded Lennard-Jones MD and Ising model simulations are better described by random walks with some memory. The seeding methods examined in chapters 4 and 5 rely on using the size of largest cluster as a reaction coordinate. This reaction coordinate is sub-optimal so, in chapter 6, we use machine learning techniques to build models to estimate the committor for the 2D Ising model. These models have a large upfront computational cost but, once trained, provide a relatively cheap estimate of the committor.

Overall, in this work we have shown that, for LJ nucleation from the melt, treating the cluster size as a Markovian random walk does not capture all of the

dynamics. We also showed that care must be taken when generating trajectories for seeding methods if we want to recover the correct free energy landscape. Finally, we used machine learning to generate models to predict the committor for a 2D Ising model based on the grid state, or simple collective variables, which could lower computational costs of calculating the committor. The results of Chapters 4 and 5 indicate that care should be taken when using seeding methods, and that models that include memory, or other reaction coordinates could give better nucleation rate estimates. Chapter 6 provides a method of accessing the usually prohibitively computationally expensive committor for use in path sampling and other calculations. These models could be extended to cover more general conditions, or more complex systems.

Appendix A

Lennard-Jones Simulation Details

This appendix covers the simulation detail for the MD simulations used in both chapters 4 and 5. All MD simulation were carried out using the LAMMPS MD engine [39], the version of LAMMPS used for all MD simulations is 2020-Mar-03. For the Lennard-Jones simulations, we use the modified LJ potential described in section 2.3.2. To implement this interaction potential in LAMMPS, we tabulate the potential in increments of $dr = 10^{-4}$ out to a maximum of $r = 3$, which is sufficient as our cut off is $r_{\text{cut}} = 2.5$. The other simulation parameters are described in [52] but also included here for convenience, table A.1. All parameters have been also been converted to reduced units.

parameter	value
pressure	-0.02
barostat damping	0.05
thermostat damping	0.05
simulation time step	0.0002
output interval	1000 steps

Table A.1: Lennard-Jones simulation parameters in reduced units.

The code below shows how we implement these parameters in our LAMMPS scripts,

```
1 pair_style table linear 10000
2 pair_coeff * * modified_lj.table MODIFIED_LJ 2.5
3
4 timestep 0.0002
5 thermo 1000
6
7 fix 1 all nph iso -0.02 -0.02 0.05
8 fix 2 all temp/csvr ${T} ${T} 0.05 ${random_seed}
```

We first define our pair style interaction, setting the interpolation to linear and the sampling rate to 10000 samples per length unit (line 1) and then importing the tabulated potential from `modified_lj.table` (line 2). This file contains the tabulated potential in the format LAMMPS expects and is generated with a simple python script that simply steps over the potential at the sampling rate. We can then define our time step and output interval (lines 4 and 5), and finally set the barostat (line 7) and thermostat (line 8) to integrate our NPT simulation. The variables `${T}` and `${random_seed}` allow us to set the thermostat temperature and a random seed for the simulation.

A.1 Seed Generation

The previous section described how we run our simulations to get our seeding trajectories, but before we can do this, we must generate our seeds. As discussed in section 4.1.1, this process can be time-consuming and is practically the hardest part of the seeding calculations. The process we go through is described in section 4.1.1, but we include it here as well for convenience.

1. Generate simulation box filled with LJ particles at the liquid density.
2. Run system above the melting temperature to melt the generated LJ crystal.
3. Rescale the velocities to the equilibration temperature T_{eq}.
4. Run the system at T_{eq} to ensure thermalisation.
5. Define the region for the seed.
6. Remove particles within the seed region.
7. Insert particles at the solid density.
8. Minimise the solid particles energies to remove overlaps.
9. Run the solid particles at T_{eq}.
10. Run the whole system at T_{eq}.

In practice, we do this in two separate simulations. First we generate the equilibrate liquid in one simulation, and then insert the seed in a second simulations. This allows us to generate all of our seeds from one liquid simulation, for a given box size, saving compute time. The implementation of this process is as follows.

A.1.1 Preparing the Liquid

The script below shows how we generate the LJ liquid box.

```
1  units lj
2  atom_style  atomic
3  atom_modify map hash
4
5  lattice fcc ${liquid_density}
6
7  region box block -${X} ${X} -${X} ${X} -${X} ${X} units lattice
8  create_box 1 box
9  create_atoms 1 box
10
11 mass 1 1.0
12
13 pair_style table linear 10000
14 pair_coeff * * modified_lj.table MODIFIED_LJ 2.5
15
16 thermo_style custom step temp press vol density ke pe v_max_n
17 timestep 0.0002
18 thermo 5000
19
20 fix 1 all nve
21 fix 2 all temp/csvr 2.0 2.0 0.05 1
22 run 250000
23 unfix 1
24 unfix 2
25
26 fix 1 all temp/rescale 1 2.0 0.5 0.01 1.0
27 run 100
28 unfix 1
29
30 thermo 1000
31 fix 1 all nve
32 fix 2 all temp/csvr 0.5 0.5 0.05 1
```

```
33 run 20000
34 unfix 1
35 unfix 2
36
37 write_restart liquid.restart
```

We begin by setting up our simulation box with the appropriate size at the liquid density, calculated from simulations at T_{eq}. Here we set the units of the system to be reduces Lennard-Jones (line 1) and set the lattice density to the liquid density (`${liquid_density}`) (line 5). We then define our simulation box size, for our convenience we centre the box on $(0, 0, 0)$ and define it in terms of a variable `${X}` (line 7). We then fill our box with LJ particles and set our interaction potential as before (lines 8–4).

We then set out simulation time step, output interval and output parameters. As we are mainly interested in the final state of these simulations, we choose a longer output interval (line 18) to reduce the computational expense. We also define what parameter will be output using the `thermo_style` command (line 16). Most of these parameters are built-in parameters in LAMMPS, however `v_max_n` is the size of largest cluster, the implementation of which is described in section 2.5.1.

After setting up the system and time steps we run the system above the melting temperature to melt the solid block. Here we combine an energy and volume conserving integration (line 20) with a `csvr` thermostat (line 21) to run under NVT conditions. We choose NVT as we inserted the particles at the liquid density for T_{eq} and if we allow the volume to change now, we will have to run for longer later so it can re-equilibrate. We choose to run at $T = 2$ as this is well above the melting temperature $T_{\text{m}} = 0.617$ [2]. Once the system has melted we rescale the particle velocities to T_{eq} (line 26) to give the thermostat less work to do. This rescales from $T = 2 \rightarrow T = T_{\text{eq}} = 0.5$ over 100 time steps.

Now we can run the system at T_{eq} (lines 31–33). At this point we, reduce the output interval (line 30) so we can see the cluster sizes change over a smaller timescale to check how liquid like the result is. We can then save the state of this simulation (line 37) so we can insert seeds of different sizes. At this point we have omitted the cluster size calculation and will discuss that later in the appendix (section A.2).

A.1.2 Inserting the Seed

This script shows the process for inserting a seed into the preformed liquid box.

```
1  read_restart liquid.restart
2
3  lattice fcc ${solid_desity}
4
5  pair_style table linear 10000
6  pair_coeff * * modified_lj.table MODIFIED_LJ 2.5
7
8  region INNER sphere 0 0 0 ${R} units lattice
9  region OUTER sphere 0 0 0 ${R} side out units lattice
10
11 delete_atoms region INNER
12 create_atoms 1 region INNER
13
14 group in region INNER
15 group out region OUTER
16
17 thermo_style custom step temp press vol density ke v_max_n
18 timestep 0.0002
19 thermo 1000
20
21 fix fix_outer out setforce 0.0 0.0 0.0
22 minimize 0.0 1.0e-8 1000 100000
23 unfix fix_outer
24
25 fix 1 in nve
26 fix 2 in temp/csvr 0.5 0.5 0.00005 1
27 run 50000
28 unfix 1
29 unfix 2
30
31 thermo 1000
32 fix 1 out nph iso -0.02 -0.02 0.05 dilate out
```

```
33 fix 2 out temp/csvr 0.5 0.5 0.05 1
34 run 40000
35 unfix 1
36 unfix 2
37
38 fix 1 all nph iso -0.02 -0.02 0.05
39 fix 2 all temp/csvr 0.5 0.5 0.05 1
40 run 40000
41 unfix 1
42 unfix 2
43
44 write_restart seed.restart
```

To insert a seed, we begin by loading the restart file for the liquid (line 1), setting the lattice density to the solid density at T_{eq} and defining our interaction potential (lines 3-6). We then define 2 regions, the `INNER` seed region, and the `OUTER` liquid region (lines 8 and 9). These regions are both defined in terms of a seed radius `${R}`, this is the reason we centred our simulation box on $(0, 0, 0)$. The `INNER` region is defined as a sphere of radius `R`, and the `OUTER` region is defined as anything outside of that. After the regions are defined, all the particles in the inner region are removed and new ones inserted (lines 11 and 12), the particles are then split into groups so they can be referred to later. By removing and reinserting the particles we now have a seed at the solid density surrounded by a liquid.

We now set the time step, output interval and output parameters (lines 17–19) and minimise the energy of the seed particles (lines 21–23). During the minimisation we set the force on the liquid particles to be zero so that it only effects the seed particles. Once the seed has been minimised to remove any accidental overlap with the liquid we run the seed particles under NVT to allow them to equilibrate at T_{eq} (lines 25–27). During this NVT, the liquid particles are kept stationary. Here the damping coefficient for the thermostat is much smaller than usual so the simulation doesn't become unstable if the particles are still close after the minimisation.

After equilibrating the seed, we run the liquid, while holding the seed stationary so it can equilibrate around the seed (lines 31-34) and then run the whole system so the boundary between liquid and seed can equilibrate (lines 38-40). Finally we

save the state of the simulation so we can run trajectories from this seed (line 44) We equilibrate each region of the system individually as because our equilibrium temperature is subcritical, $T_{\text{eq}} < T_c$, we expect our seed to grow anytime the liquid region is moving.

A.2 Cluster Size Calculations

To calculate the size of largest cluster in our system we use the method described by ten Wolde in [23] which we discussed in section 2.5.1. The script below demonstrates how we implement this in our LAMMPS scripts.

```
1  compute q6 all orientorder/atom degrees 1 6 components 6 nnn
       NULL cutoff 1.432
2
3  compute coord_number all coord/atom orientorder q6 0.5
4  variable is_solid atom 'c_coord_number >= 8'
5  group solid dynamic all var is_solid every 1000
6
7  compute cluster solid cluster/atom 1.432
8  compute clus_chunks solid chunk/atom c_cluster
9  compute size_chunks solid property/chunk clus_chunks count
10
11 variable max_n equal max(c_size_chunks)
```

First we compute the Steinhardt parameters, $\mathbf{q}_6$, for each particle (line 1), we use a cut-off of 1.432 as that is the same as [52]. Those $\mathbf{q}_6$ vectors can then be used to calculated the coordination number of each particle (line 3), that is, how many "connections" it has. The coordination number is then use to separate the particles into solid and liquid (line 4). If a particle has a coordination number $C >= 8$ then it is solid, otherwise it is liquid. The solid group is set to dynamic so that it updates before every output interval. We then perform a clustering sweep on the particles, assigning them as cluster id number (line 7), splitting them into groups based on their cluster number (line 8), and counting the number of particles in each cluster (line 9). The cut-off range for the clustering is the same as for the $\mathbf{q}_6$ calculations. After computing the size of all clusters we then select the maximum to output from our simulations (line 11). This is quite an involved process and is the most computationally intensive part of our seeding simulations.

Bibliography

[1] Gregg T. Beckham and Baron Peters. Optimizing nucleus size metrics for liquid-solid nucleation from transition paths of near-nanosecond duration. *Journal of Physical Chemistry Letters*, 2(10):1133–1138, 2011.

[2] Jorge R. Espinosa, Carlos Vega, Chantal Valeriani, and Eduardo Sanz. Seeding approach to crystal nucleation. *Journal of Chemical Physics*, 144(3):034501, jan 2016.

[3] Richard Rhodes. Ten things we need to know about ice and snow. *Nature*, 494:27–29, 2013.

[4] E. G. Hammerschmidt. Formation of Gas Hydrates in Natural Gas Transmission Lines. *Industrial and Engineering Chemistry*, 26(8):851–855, 1934.

[5] E. Dendy Sloan. Fundamental principles and applications of natural gas hydrates. *Nature*, 426(6964):353–359, 2003.

[6] Jason R. Cox, Lori A. Ferris, and Venkat R. Thalladi. Selective growth of a stable drug polymorph by suppressing the nucleation of corresponding metastable polymorphs. *Angewandte Chemie - International Edition*, 46(23):4333–4336, 2007.

[7] Andrew P. Evan, Elaine M. Worcester, Fredric L. Coe, James Williams, and James E. Lingeman. Mechanisms of human kidney stone formation. *Urolithiasis*, 43(1):19–32, 2014.

[8] Dominic M Walsh, Aleksey Lomakin, George B Benedek, Margaret M Condron, and David B Teplow. Amyloid beta-Protein Fibrillogenesis. *The Journal of Biological Chemistry*, 272(35):22364–22372, 1997.

[9] G. N. Rimondino, E. Miceli, M. Molina, S. Wedepohl, S. Thierbach, E. Rühl, M. Strumia, M. Martinelli, and M. Calderón. Rational design of dendritic thermoresponsive nanogels that undergo phase transition under endolysosomal conditions. *Journal of Materials Chemistry B*, 5(4):866–874, 2017.

[10] Gabriele C. Sosso, Ji Chen, Stephen J. Cox, Martin Fitzner, Philipp Pedevilla, Andrea Zen, and Angelos Michaelides. Crystal Nucleation in Liquids: Open Questions and Future Challenges in Molecular Dynamics Simulations. *Chemical Reviews*, 116(12):7078–7116, 2016.

[11] Thomas Palberg. Crystallization kinetics of colloidal model suspensions: Recent achievements and new perspectives. *Journal of Physics Condensed Matter*, 26(33):333101, 2014.

[12] Roy J. Glauber. Time-dependent statistics of the Ising model. *Journal of Mathematical Physics*, 4(2):294–307, 1963.

[13] Kyozi Kawasaki. Diffusion constants near the critical point for time-dependent ising models. I. *Physical Review*, 145(1):224–230, 1966.

[14] Mark E. Tuckerman. *Statistical mechanics: theory and molecular simulation.* Oxford University Press, 2010.

[15] Shuichi Nosé. A unified formulation of the constant temperature molecular dynamics methods. *The Journal of Chemical Physics*, 81(1):511–519, 1984.

[16] William G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31(3):1695–1697, 1985.

[17] Glenn J. Martyna, Michael L. Klein, and Mark Tuckerman. Nosé-Hoover chains: The canonical ensemble via continuous dynamics. *The Journal of Chemical Physics*, 97(4):2635–2643, 1992.

[18] Giovanni Bussi, Davide Donadio, and Michele Parrinello. Canonical sampling through velocity rescaling. *Journal of Chemical Physics*, 126(1):014101, 2007.

[19] Jeremy Q. Broughton and George H. Gilmer. Molecular dynamics investigation of the crystal-fluid interface. I. Bulk properties. *The Journal of Chemical Physics*, 79(10):5095–5104, 1983.

[20] Baron Peters. *Reaction Rate Theory and Rare Events.* Elsevier B.V., 2017.

[21] M. F. Riley, M. P. Hobson, and S. J. Bence. *Mathematical Methods For Physics and Engineering.* Cambridge University Press, 2006.

[22] Paul J. Steinhardt, David R. Nelson, and Marco Ronchetti. Bond-orientational order in liquids and glasses. *Physical Review B*, 28(2):784–805, 1983.

[23] Pieter Rein Ten Wolde, Maria J. Ruiz-Montero, and Daan Frenkel. Simulation of homogeneous crystal nucleation close to coexistence. *Faraday Discussions*, 104:93–110, 1996.

[24] Baron Peters and Bernhardt L. Trout. Obtaining reaction coordinates by likelihood maximization. *Journal of Chemical Physics*, 125(5):054108, 2006.

[25] Weinan E, Weiqing Ren, and Eric Vanden-Eijnden. Transition pathways in complex systems: Reaction coordinates, isocommittor surfaces, and transition tubes. *Chemical Physics Letters*, 413(1-3):242–247, 2005.

[26] Alessandro Laio and Michele Parrinello. Escaping free-energy minima. *Proceedings of the National Academy of Sciences of the United States of America*, 99(20):12562–12566, oct 2002.

[27] G M Torrie and J P Valleau. Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling. *Journal of Computational Physics*, 23(2):187–199, 1977.

[28] Christoph Dellago, Peter G. Bolhuis, and Phillip L. Geissler. *Advances in Chemical Physics - Chapter 1 - Transition Path Sampling.* John Wiley & Sons, Inc, 2002.

[29] P. G. Bolhuis and C. Dellago. Practical and conceptual path sampling issues. *European Physical Journal: Special Topics*, 224(12):2409–2427, 2015.

[30] Rosalind J. Allen, Chantal Valeriani, and Pieter Rein Ten Wolde. Forward flux sampling for rare event simulations. *Journal of Physics Condensed Matter*, 21(46):463102, 2009.

[31] Nils B. Becker, Rosalind J. Allen, and Pieter Rein Ten Wolde. Non-stationary forward flux sampling. *Journal of Chemical Physics*, 136(17):1–18, 2012.

[32] Joshua T. Berryman and Tanja Schilling. Sampling rare events in nonequilibrium and nonstationary systems. *Journal of Chemical Physics*, 133(24):244101, 2010.

[33] Brandon C. Knott, Valeria Molinero, Michael F. Doherty, and Baron Peters. Homogeneous nucleation of methane hydrates: Unrealistic under realistic conditions. *Journal of the American Chemical Society*, 134(48):19544–19547, 2012.

[34] Gerhard Hummer. From transition paths to transition states and rate coefficients. *Journal of Chemical Physics*, 120(2):516–523, 2004.

[35] Daniele Moroni, Pieter Rein Ten Wolde, and Peter G. Bolhuis. Interplay between structure and size in a critical crystal nucleus. *Physical Review Letters*, 94(23):1–4, 2005.

[36] J. Kuipers and G. T. Barkema. Limitations of a Fokker-Planck description of nucleation. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 82(1):1–5, 2010.

[37] Seunghwa Ryu and Wei Cai. Numerical tests of nucleation theories for the Ising models. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 82(1):1–14, 2010.

[38] V. A. Shneidman, K. A. Jackson, and K. M. Beatty. On the applicability of the classical nucleation theory in an Ising system. *Journal of Chemical Physics*, 111(15):6932–6941, 1999.

[39] Steve Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117(1):1–19, 1995.

[40] Massoud Kaviany. *Heat Transfer Physics*. Cambridge University Press, second edi edition, 2014.

[41] Mathilde Bugel and Guillaume Galliero. Thermal conductivity of the Lennard-Jones fluid: An empirical correlation. *Chemical Physics*, 352(1-3):249–257, 2008.

[42] E. R. Cowley. Some Monte Carlo calculations for the Lennard-Jones solid. *Physical Review B*, 28(6):3160–3163, 1983.

[43] Bruce J Palmer. Calculation of thermal-diffusion coefficients from plane-wave fluctuations in the heat energy density. *Physical Review E*, 49(3):2049–2057, 1994.

[44] David Quigley. GPU accelerated 2D Ising model, 2020. (https://github.com/dquigley-warwick/GPU_2DIsing).

[45] Jonathan Gross, Johannes Zierenberg, Martin Weigel, and Wolfhard Janke. Massively parallel multicanonical simulations. *Computer Physics Communications*, 224:387–395, 2018.

[46] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[47] Ao Ma and Aaron R. Dinner. Automatic method for identifying reaction coordinates in complex systems. *Journal of Physical Chemistry B*, 109(14):6769–6779, 2005.

[48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfittin. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[49] Jürgen Franke and Michael H. Neumann. Bootstrapping neural networks. *Neural Computation*, 12(8):1929–1949, 2000.

[50] D. A. Nix and A. S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, pages 55–60 vol.1, 1994.

[51] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *33rd International Conference on Machine Learning, ICML 2016*, 3:1651–1660, 2016.

[52] Jorge R. Espinosa, Carlos Vega, Chantal Valeriani, and Eduardo Sanz. The crystal-fluid interfacial free energy and nucleation rate of NaCl from different simulation methods. *Journal of Chemical Physics*, 142(19):194709, 2015.